UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
DE SISTEMAS INFORMÁTICOS

APLICACIÓN ANDROID DE TWITTER EXTENDIDA CONSUMIENDO UN SERVICIO REST Y SUS PRUEBAS

PROYECTO FINAL DE MÁSTER

TITULACIÓN:

MÁSTER EN INGENIERÍA WEB

AUTORES:

KATHERIN MOSCOSO LEÓN

SALVADOR JOSÉ GARCÍA ALCALÁ

DIRECTOR:

FRANCISCO JAVIER GIL RUBIO

ÍNDICE DE CONTENIDOS

1 INTRODUCCIÓN 6

1.1 Abstract 7

2 OBJETIVOS 8

2.1 Lista de Requisitos 8

3 ESTUDIO DEL ESTADO DEL ARTE 9

3.1 Dispositivos Móviles 9

3.2 Sistemas Operativos Móviles 10

3.2.1 iOS 10

3.2.2 Android 11

3.2.3 Windows Phone 11

3.2.4 Firefox OS 12

3.2.5 Blackberry 13

3.2.6 Symbian 13

3.2.7 Ubuntu Touch 14

4 TECNOLOGÍA 15

4.1 Android 15

4.1.1 Versiones Android 16

4.2 REST 17

4.3 GitHub 18

5 FUTURO DE ANDROID 20

5.1 Android M 20

5.1.1 Fecha de Lanzamiento 20

5.1.2 Versión Preview de Android M 21

5.1.3 Fecha de Lanzamiento de Android M para el Resto de Terminales 21

5.1.4 Funciones y Novedades 21
5.1.5 ¿Cuál será el nombre de Android M? 25
5.2 ¿Podrá ser Dart y no Java el Futuro de Android? 26
6 ANDROID STUDIO 28
6.1 Estructura de un Proyecto en Android Studio 28
6.1.1 Directorios de un Proyecto en Android Studio 29
6.1.2 ¿Qué es el Android Manifest? 31
6.1.3 ¿Qué utilidad tiene el archivo strings.xml? 33
6.1.4 ¿Qué hay dentro de la carpeta layout? 35
6.1.5 La carpeta mipmap en Android Studio 37
6.1.6 Comprendiendo el propósito de la clase R.java 39
6.1.7 La carpeta java de un proyecto en Android Studio 41
6.2 Estados de una Actividad Android 42
7 MATERIAL DESIGN 44
7.1 ¿En qué se basa Material Design? 45
7.2 Material Design, un Diseño Adaptado Para Todo Tipo de Pantallas 47
7.3 Claves de Material Design que hemos Aplicado 48
8 METODOLOGÍAS 49
8.1 Metodología de Gestión 49
8.2 Metodología de Desarrollo 49
9 ARQUITECTURA 50
9.1 Patrón Diseño 51
10 DESARROLLO 54
10.1 Diseño 54
10.2 Implementación 56
10.2.1 Autenticación en Twitter con Fabric 56
10.2.2 Realizar llamadas a la API de Twitter con Twitter4J 59

10.2.3 Uso de tabs 60
10.2.4 Fragmentos 62
10.2.5 Adapters 65
10.2.6 Asyntask 65
10.2.7 Buenas prácticas implementadas 66
11 PRUEBAS 68
12 GOOGLE PLAY 70
12.1 Clasificación de Aplicaciones 72
12.2 Interfaz 73
12.3 Desarrolladores 73
12.4 Disponibilidad para Desarrolladores 74
12.5 Competencia y Aliados 74
13 Publicación en Google Play 75
13.1 Versión de la Aplicación 75
13.2 Crear Firma Digital de la Aplicación 75
13.3 Exportar la Aplicación 76
13.4 Consola de Desarrollador 77
13.5 Publicación de la Aplicación 77
14 CONCLUSIONES 78
15 LÍNEAS FUTURAS 79
16 Referencias 80
17 BIBLIOGRAFÍA 81

LISTA DE FIGURAS

Figura 1: Sistemas Operativos para Telefonos Moviles (Blog del CEU) 10
Figura 2 15
Figura 3 16
Figura 4 17
Figura 5: Versiones de Android 20
Figura 6: Control de Permisos Granulares 22
Figura 7: Personalización de Parámetros Rápidos 23
Figura 8: Cable USB Type C 24
Figura 9: Posible nombre de Android M 25
Figura 10: Logo de Dart 26
Figura 11: Logo de Android Studio 28
Figura 12: Estructura de un Proyecto en Android Studio 29
Figura 13: Estructura de la Carpeta APP 30
Figura 14: Estructura de la Carpeta SRC (Source) 30
Figura 15: Archivo Android Manifest 31
Figura 16: Ruta del Archivo Strings 33
Figura 17: Archivo Strings 34
Figura 18: Estructura de la Carpeta Layout 35
Figura 19: Archivo Activity_main 36
Figura 20: Contenido de la Carpeta Mipmap 37
Figura 21: Diferentes Densidades de Pantalla 38
Figura 22: Localización de la Clase R de Java 39
Figura 23: Archivo R de Java 40
Figura 24: Estructura de la Carpeta Java 41
Figura 25: Archivo TweetDetailActivity 42
Figura 26: Estados de una Actividad Android 42
Figura 27: Relieve en Material Design 44
Figura 28: Elementos Ordenados 45
Figura 29: Luz y Sombras en Android 46
Figura 30: Resoluciones de Pantalla 47
Figura 31: Colores Seleccionados para la Aplicación 48

Figura 32 50
Figura 33 51
Figura 34: Boceto de la Aplicación 54
Figura 35: Captura del Diseño de la Aplicación 55
Figura 36 57
Figura 37 57
Figura 38 60
Figura 39 61
Figura 40 62
Figura 41 63
Figura 42 64
Figura 43 68
Figura 44 68
Figura 46: Logo de Google Play 70
Figura 47: Captura de Pantalla de la Firma de la Aplicación 76

1 INTRODUCCIÓN

En este proyecto se pretende realizar una aplicación cliente de Twitter con Android, que permita al usuario realizar tareas sencillas interactuando con la aplicación de Twitter. Este proyecto pretende ser una guía básica para explicar cómo realiza la conexión con una aplicación con Twitter y explicar las ventajas del uso de Android. Por un lado, la tecnología Android es una las tecnologías más punteras en el desarrollo de móviles en la actualidad, actualmente es líder en el mercado en la venta de dispositivos móviles. Por otro lado, Twitter es una de las redes sociales más usadas en la actualidad, no sólo por el público en general, si no que muchas empresas ofrecen información y soporte al usuario por este medio. La API de Twitter, al igual que Android, es de libre distribución, por lo que resulta sencillo encontrar información acerca de cómo implementarlos juntos.

1.1 Abstract

The final project consist on development a Twitter [1] client application using Android [2] as a development tool. The major purpose of this document is to implement a guide for future developers, this guide explain how to make an authentication with Twitter and explain the main advantages of Android as a development tool for mobile device. On one hand, Android is the market leader in mobile device and is the favorite operation system for low cost device, which mean a huge amount of device that use it. On the other hand, Twitter is one of the most visited web pages around the world. User mainly user twitter for publish a Twitter but many companies use its API to provide service through Twitter such as information or authentication for their own application. Twitter and Android are open sources projects, this is a major advantages because there is more information in the network about how to implement this technologies.

The previous steps of development and codification was looking for update of this technologies. One of the main aspect is the use of Android Studio which is the official IDE by Google.

2 OBJETIVOS

El objetivo principal del presente proyecto se centra en el desarrollo de una aplicación Android de Twitter extendida que funciona consumiendo el servicio API REST del propio Twitter.

Otro de los principales objetivos que nos hemos marcado ha sido el de seguir las últimas directrices en programación Android, para poder explotar las últimas funcionalidades de este Sistema Operativo Móvil. Para ello, hemos utilizado Android Lollipop y hemos realizado una aplicación Material Design.

2.1 Lista de Requisitos

- Iniciar Sesión: La aplicación requerirá que el usuario se autentique a través de su cuenta de Twitter.
- Tuitear: El sistema debe permitir que el usuario pueda escribir y publicar un Tweet.
- Responder Tweet: El sistema debe permitir que el usuario pueda responder un Tweet.
- Retuitear Tweet: El usuario debe poder retuitear un tweet
- Marcar Favorito: El usuario debe poder marcar un tweet como favorito
- Actualización Timeline: La vista del Timeline se tiene que poder actualizar para que el sistema te muestre los últimos tweet cuando actualizas la vista
- Detalle del Tuit: El sistema debe poder mostrarle al usuario una pantalla de detalle de cada uno de los tuits
- Foto Usuarios: La aplicación debe mostrar las fotografías de perfil de los usuarios que han escrito un tuit
- Carga infinita de Tuits: La aplicación debería dejar cargar la pantalla del Timeline para poder visualizar todos los tuits que se han escrito a lo largo del tiempo.

3 ESTUDIO DEL ESTADO DEL ARTE

3.1 Dispositivos Móviles

Según el VI Estudio Anual de Mobile Marketing (último disponible hasta la fecha), realizado por la Comisión de Mobile lab Spain junto con The Cocktail Analysis y en el que se analizan los hábitos de los usuarios de dispositivos móviles en España, el 60% de la población se conecta a internet y el 52% ya dispone de un Smartphone.

El tiempo de conexión de media en Móvil entre gente que se conecta a diario a la Red, que representan el 81% de los usuarios de Internet, consumen 2 horas y 32 minutos diarios. Mientras que el uso desde la Tablet entre los usuarios diarios, que supone el 43% de los internautas, están conectados al día casi 2 horas.

Con respecto al uso del email: El 83% de los usuarios accede a su email desde el móvil al menos una vez a la semana (78% en 2013) y demanda recibir emails personalizados según sus gustos y preferencias. El 50% de los que acceden semanalmente al correos desde el móvil consulta marcas y tiendas al menos una vez al día y el 30% de las personas vuelve a abrir los email de una marca.

En lo relacionado con los anuncios de las marcas, lo que quiere el consumidor son Descuentos y más información. Una vez que se hace click, 1 de cada 2 personas muestra alta satisfacción tras la visita.

En el terreno del uso de Apps: WhatsApp incrementa su ventaja como mensajería instantánea, con un 72% de los usuarios de que utilizan dispositivos móviles. Facebook y Twitter siguen liderando las Apps de Redes Sociales, subiendo Instagram y Youtube. En lo referente a las Tablet, Facebook sigue siendo el rey y los juegos “juegan” un papel importante.

La Televisión y los dispositivos móviles cada vez van más de la mano. El móvil se convierte en la segunda pantalla, el 90% de los usuarios de los internautas móvil lo usa mientras ve la TV, y la mitad lo hace con frecuencia. Las actividades favoritas mientras se ve la televisión están relacionadas con las Redes Sociales y el correo electrónico.

El Smartphone en la decisión de compra cada vez tiene más protagonismo:

- El 90% utiliza el móvil a la hora de decidirse.
- En un 80% para buscar información sobre las características que se está comprando, precios, artículos similares, opiniones de otras personas, etc.
- El 45% ha efectuado alguna compra a través del dispositivo móvil.
- Lo que motivó a la compra: La búsqueda (Search), los anuncios (display) y las Apps.

3.2 Sistemas Operativos Móviles

Los sistemas operativos usados para los teléfonos móviles, celulares o smartphone son muchos, pero hay 2 que son los principales y que ocupan casi todo el mercado de la telefonía móvil: Android e iOS. Seguidos pero con mucha diferencia estarían Symbian, Blackberry OS, Windows Phone, Firefox OS y Ubuntu Touch.

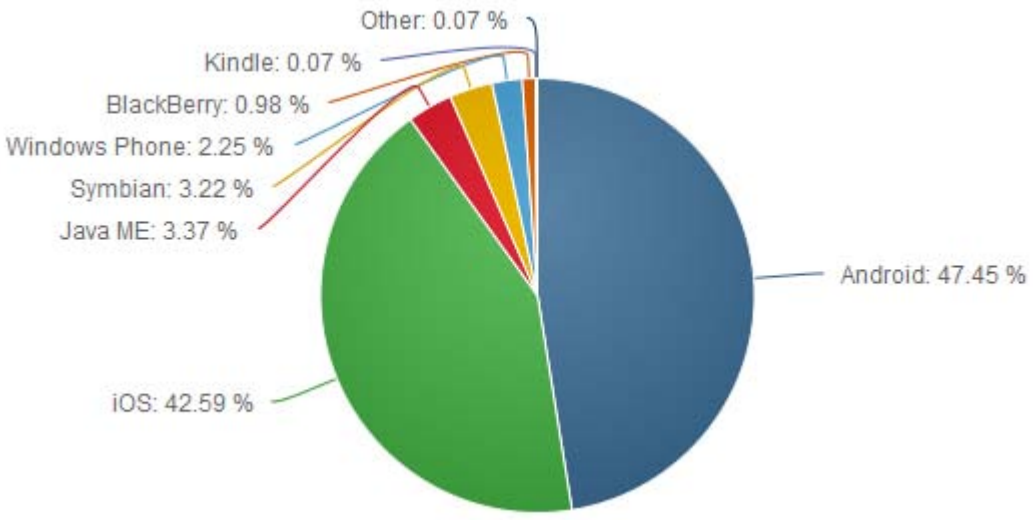


FIGURA 1: SISTEMAS OPERATIVOS PARA TELEFONOS MOVILES (BLOG DEL CEU)

3.2.1 iOS

La última versión del sistema operativo del iPhone es la 8. Los nuevos terminales de la compañía de Cupertino, el iPhone 6 y 6 Plus, están diseñados para sacarle todo el partido, aunque desde el iPhone 5S en adelante se puede disfrutar de esta versión de iOS. Las

principales características que trae son: una mayor integración con OSX y la extensión de su ecosistema hacía los wearables y el internet de las cosas con el healthkit y homekit respectivamente.

Lo que caracteriza a iOS frente a otros es que es un sistema operativo cerrado. Apple no permite que se modifiquen características internas del sistema más allá de las limitadas opciones que da en los ajustes. Un sistema cerrado permite, sin embargo, ofrecer siempre una experiencia más estable y segura, tal y como diseñó el fabricante en un principio. Sin embargo a muchos usuarios, que buscan una mayor personalización, se les pueden quedar cortas las opciones que le da Apple.

Por otro lado, como también suele ser habitual en los productos de la empresa, no se licencia a terceros por lo que tan sólo los iPhone disponen de este sistema operativo.

3.2.2 Android

El sistema operativo número uno en cuanto a popularidad. Con una cuota de mercado cercana al 85%, el sistema operativo de Google se caracteriza por ser abierto y disponible para cualquier fabricante interesando en utilizarlo para sus dispositivos móviles.

Esta disponibilidad ha creado sin embargo una gran fragmentación, pudiéndose encontrar innumerables dispositivos de miles de formas y funcionalidades con todas las versiones de Android existentes. Además la posibilidad de que cada fabricante incluya su propia capa sobre el original, propicia que la experiencia de usuario no sea siempre la deseada por Google y las actualizaciones tarden en llegar.

Una penetración de mercado tan grande, ha propiciado por otro lado, que aunque en un primer momento iOS fuera el más popular de los SO para los desarrolladores, cada vez más, estos dedican grandes esfuerzos a diseñar sus apps para los usuarios de Android.

3.2.3 Windows Phone

Microsoft está realizando un gran esfuerzo financiero para posicionar Windows Phone como una tercera opción interesante para los consumidores después de que llegara tarde a la fiesta de los smartphones. Su alianza con Nokia y su posterior compra le ha

ayudado a darse a conocer mejor e ir arañando cuota de mercado a los dos líderes. Los últimos datos hablan de un 2,5% a nivel mundial.

Con un diseño radicalmente distinto a las dos opciones ya comentadas, Windows Phone destaca por su pantalla de inicio personalizable que ofrece las notificaciones de las apps de una manera sencilla y limpia. Además ofrece una experiencia de usuario muy buena independientemente del tipo y gama de terminal en que se esté usando.

Aunque con menos apps disponibles que en Android y iOS, Windows Phone 8.1, cuenta ya con más de 300.000 apps en su tienda, además de ofrecer aplicaciones propias de la compañía como Skype, OneDrive o Xbox Live.

3.2.4 Firefox OS

Un sistema operativo basado en HTML5 con núcleo Linux, de código abierto. Desarrollado por Mozilla Corporation con apoyo de empresas como Telefónica. El sistema operativo está basado en Linux y usa la tecnología de Mozilla, Gecko. Se basa en estándares abiertos como por ejemplo HML5, CSS3 y JavaScript.

Pensado para ser un sistema operativo realmente abierto, a diferencia de Android, donde Google controla ciertos aspectos del sistema. Esta característica, permite a Firefox OS llegar a cubrir el nicho de mercado de la gama baja con mayor facilidad que Android. Movistar ya lanzó hace más de un año los primeros smartphone con este sistema operativo en España y Latinoamérica.

Entre las interesantes características de este sistema operativo abierto están las aplicaciones web y pueden ser de dos tipos diferentes: aplicaciones de servidor o empaquetadas. A diferencia de los SO ya comentados, en este caso, las apps de servidor, corren vía web, es decir son páginas webs con la apariencia de aplicaciones y sin conexión a internet no es posible acceder a estas. Las aplicaciones empaquetadas necesitan la descarga de un paquete comprimido y se cargan desde la fuente local cada vez que se accede a la aplicación.

3.2.5 Blackberry

Blackberry, anteriormente conocida como RIM, no está pasando por sus mejores momentos. Al igual que le pasó a Nokia, el cambio de paradigma en los smartphones les pilló con el pie cambiado. Acostumbrado a ofrecer terminales con teclado físico, el paso a las pantallas táctiles se les atragantó. Sin embargo, los esfuerzos realizados por la compañía canadiense para recuperar el terreno perdido han sido grandes y en el año 2012 lanzaron su órdago con un renovado sistema operativo, el Blackberry 10. Aun así, los últimos estudios sobre cuota de mercardo lo dejan en tan sólo un 0,5% mundial.

Blackberry 10 tiene una interfaz más fluida, un teclado inteligente y táctil más depurado y otra serie de opciones que lo acercan a las de la competencia. Al igual que con iOS, el SO es software propietario y solamente los teléfonos de la compañía llevan su sistema instalado.

3.2.6 Symbian

- Posee un eficiente uso de todos los recursos de la máquina (especialmente de la batería, la memoria RAM y la ROM).

- La paginación bajo demanda permite un mejor aprovechamiento de la memoria RAM de los dispositivos ya que sólo se carga en memoria la "página" que se va a ejecutar.

- El sistema posee componentes que permiten el diseño de aplicaciones multiplataforma, o sea, diferentes tamaños de pantalla, color, resolución, teclados, etc.

- Permite la conectividad con diferentes dispositivos a través de Bluetooth.

3.2.7 Ubuntu Touch

Se trata de otro sistema operativo basado en Linux pero en esta ocasión bajo la famosa firma Ubuntu. Presentado en el 2013, se trata de un proyecto de Canonical. En la actualidad varias empresas están desarrollando terminales para este sistema operativo, entre ellas la española Bq.

Ubuntu Touch utiliza las mismas tecnologías de la versión de escritorio, por lo que ambas comparten apps sin problemas de compatibilidad. Dispone también de algunas de las aplicaciones más populares como Facebook y Youtube.

4 TECNOLOGÍA

4.1 Android

Android es un sistema operativo basado en Linux de código abierto, por lo que, cualquier persona puede acceder de forma gratuita. Fue diseñado para dispositivos táctiles como pueden ser tabletas, Smartphone, entre otros. El uso de Android tiene algunas ventajas, de las cuales podemos destacar las siguientes:

- Es el sistema operativo más usado actualmente, según el estudio de Gartner [1] a venta de Smartphone con sistema operativo Android ocupo el primer lugar con un porcentaje del 80,7.

Table 3

Worldwide Smartphone Sales to End Users by Operating System in 2014 (Thousands of Units)

Operating System	2014 Units	2014 Market Share (%)	2013 Units	2013 Market Share (%)
Android	1,004,675	80.7	761,288	78.5
iOS	191,426	15.4	150,786	15.5
Windows	35,133	2.8	30,714	3.2
BlackBerry	7,911	0.6	18,606	1.9
Other OS	5,745	0.5	8,327	0.9
Total	**1,244,890**	**100.0**	**969,721**	**100.0**

Source: Gartner (March 2015)

Worldwide mobile phone sales to end users totaled nearly 1.9 billion units in 2014, a 3.9 percent increase from the same period in 2013 (see Table 4). All regions recorded growth in 2014, except Japan and Western Europe, which recorded declines of 2.8 percent and 9.1 percent, respectively.

FIGURA 2

- Sistema gratuito con más de 1.000.000 aplicaciones, que sea de distribución gratuita hace que la comunidad de desarrolladores sea mayor que la de otros sistemas operativos en móviles. De esta forma, encontrar información y tutoriales que permitan el desarrollo del proyecto es más fácil.

- Coste de desarrollo inferior al de su directo competido iOS, ya que, la licencia para publicar aplicaciones es inferior (100€ en iOS cada año, 30€ en Android para toda la vida) y, no es necesario tener un ordenador Mac para desarrollar la aplicación.

4.1.1 Versiones Android

Android tiene un inconveniente destacable, muchas versiones del sistema operativo. Desde la versión primera (Android 1.0 Apple Pie) hasta la última actualmente (la 5.0, Lollipop), han pasado 14 versiones diferentes de Android entre ellas [2]. Según publica Android [3] Mayo de 2015 el porcentaje de sus usuarios que utiliza cada una de las versiones diferentes de su sistema operativo se puede observar en **¡Error! No se encuentra el origen de la referencia.**

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	5.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.3%
4.1.x	Jelly Bean	16	15.6%
4.2.x		17	18.1%
4.3		18	5.5%
4.4	KitKat	19	39.8%
5.0	Lollipop	21	9.0%
5.1		22	0.7%

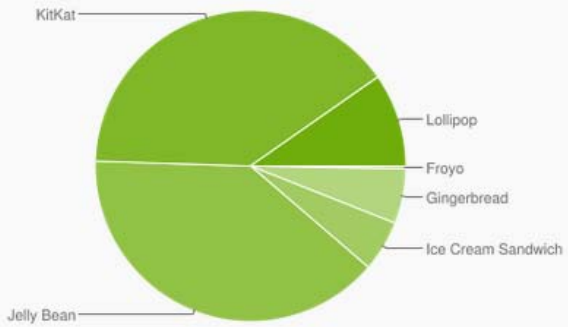
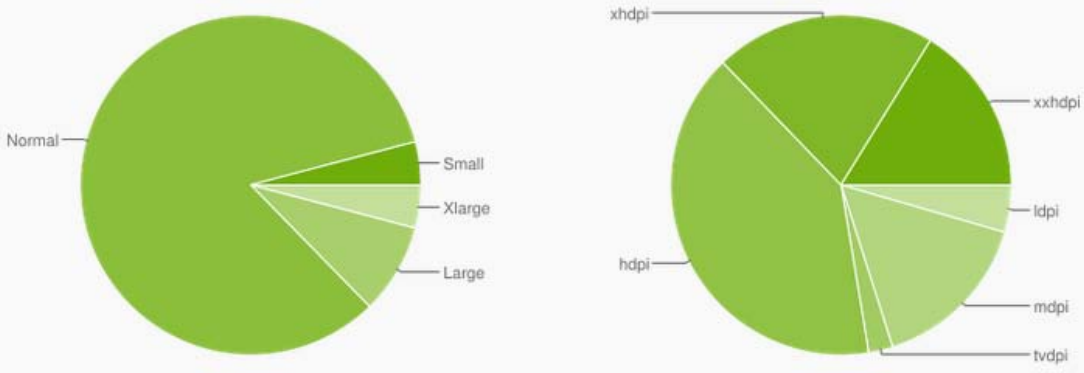


FIGURA 3

Teniendo en cuenta la gráfica se ha considerado realizar la aplicación a partir de la versión 16 Jelly Bean. A partir de esta gráfica se tiene una cobertura del 88.7 % del mercado

Uno de los factores relevantes cuando se desarrolla una aplicación móvil es el tamaño de las pantallas, actualmente en el mercado existe un catálogo muy variable con respecto al tamaño y resolución de pantallas. La resolución se mide en DPI (puntos por pulgada) en el siguiente orden: ldpi, mdpi, tvdpi, hdpi, xhdpi y xxhdpi. A continuación se muestra un gráfico que publica Android [3] referente a la distribución de los usuarios de Android en cuanto a la resolución de pantallas:

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	4.1%						4.1%
Normal		7.6%	0.1%	39.5%	19.8%	16.2%	83.2%
Large	0.4%	4.8%	2.2%	0.6%	0.6%		8.6%
Xlarge		3.2%		0.3%	0.6%		4.1%
Total	4.5%	15.6%	2.3%	40.4%	21.0%	16.2%	



Data collected during a 7-day period ending on May 4, 2015.
Any screen configurations with less than 0.1% distribution are not shown.

FIGURA 4

Para poder obtener resultados aceptable en distintos dispositivos se han incluido gráficos con diferentes densidades de pantalla para adaptar la aplicación a todo tipo de dispositivo Android.

4.2 REST

La tecnología REST (Representational State tranfer) fue creado enfocado a proporcionar un modelo arquitectónico para construir servicios web escalables [6].Las claves de esta tecnología ayudan a mejorar el rendimiento, escabilidad de las interfaces, modificación de los componentes, visibilicdad entre los componentes de la aplicación, portabilidad de los componente, moviendo el código con los datos, y por último la seguridad. La tecnología Rest se basa en las siguientes claves:

- ***Protocolo cliente/servidor sin estado:*** cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes.

- ***Conjunto de operaciones que se aplican a todos los recursos de información:*** HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE.
- ***Sintaxis universal para identificar los recursos:*** En un sistema REST, cada recurso se puede direccionar únicamente por su URI
- ***Hipermedios***: La representación de este estado en un sistema REST son típicamente HTML o XML.
- ***Sin estado:*** Cuando se realiza la comunicación entre cliente servidor no se almacena ninguna información entre ellos. Cada petición del cliente contiene toda la información necesaria para realizar la petición, siendo el cliente el que guarda los datos de sesión.
- ***Cacheable:*** Los clientes pueden cachear las respuestas.
- ***En capas:*** Un cliente saber si está conectado directamente al servidor final, o a un intermediario. Los servidores intermediarios pueden mejorar la escalabilidad del sistema, permitiendo el equilibrio de carga, proporcionando caches compartidas e implementándola seguridad.

4.3 GitHub

Para desarrollar el código del proyecto se hizo uso de un controlador de versiones Git, en este caso Github [7]. Github es un controlador de versión de libre distribución dónde el código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago o asociando una cuenta oficial de estudiantes a la cuenta. En este caso hemos creado un proyecto público, ya que, creemos que puede ser de ayuda para futuros desarrolladores tener disponible este código en la web.

Las ventajas de usar Github son las siguientes:

- Wiki para cada proyecto
- Página web para cada proyecto

- Gráfico para ver cómo los desarrolladores trabajan en sus repositorios y bifurcaciones del proyecto
- Funcionalidades como si se tratase de una red social, como por ejemplo: seguidores.
- Bueno para trabajo colaborativo entre programadores.

El proyecto se desarrolló en un sola rama, máster. Para subir la información al repositorio el programador tiene que realizar primero un update del proyecto y a continuación realizar e commit.

5 FUTURO DE ANDROID

5.1 Android M

Ha pasado menos de un año desde que salió a la luz Android L, pero para Google eso ha sido tiempo suficiente para presentar su nueva versión. Android M Preview Developer ya es una realidad.

Android M no supone un gran cambio en términos de diseño, pero sí tenemos un par de novedades interesantes, sobre todo en experiencia de usuario y en batería.

FIGURA 5: VERSIONES DE ANDROID

5.1.1 Fecha de Lanzamiento

La fecha de lanzamiento de Android M aún no está confirmada, pero todo apunta a que Google hará disponible esta nueva versión después de verano, junto a la salida del próximo teléfono Nexus, que vendrá con la nueva versión incorporada. La versión anterior, Lollipop también se dio a conocer durante el Google I/O (esta vez el de 2014) y salió oficialmente a la luz junto con el Nexus 6 en octubre, por lo que para el Android M también se calculan aproximadamente esas fechas.

5.1.2 Versión Preview de Android M

Al final de la conferencia de Keynote durante el evento de Google I/O, Google puso a disponibilidad de los desarrolladores la versión preview de Android M para algunos terminales de la gama Nexus. Entre los afortunados se encuentran el Nexus 5, 6, 9 y Player.

En Android SDK Manager de Android Studio ya se encuentra disponible el SDK Preview de Android M junto con las imágenes para poder emularlo.

5.1.3 Fecha de Lanzamiento de Android M para el Resto de Terminales

Aparte de los dispositivos Nexus, los únicos terminales cuya actualización a Android M ha sido confirmada han sido el HTC One M9 y el M9+. Suponemos que los actuales buques insignia de los grandes fabricantes contarán con su actualización a M relativamente pronto, desde el momento de su salida.

5.1.4 Funciones y Novedades

Durante su presentación oficial, Google mostró un número considerable de novedades que aparecerán en su nueva actualización. A continuación, detallamos todas las nuevas funciones que se podrán ver en Android M.

a) ***Soporte nativo de microSD:*** Tras años de peticiones de usuarios, Google por fin ha accedido a diseñar un soporte nativo para tarjetas microSD, que podrán interactuar con el teléfono como si se tratara de una extensión de la propia memoria interna, permitiendo incluso instalar aplicaciones externas dentro de la tarjeta microSD extraíble. Además, esta nueva función hará que el menú sea cambiado ligeramente para que, si tenemos aplicaciones guardadas en la microSD, podamos ver el menú fragmentado entre lo almacenado en el teléfono y lo que está en la tarjeta de memoria.

b) ***Control de permisos granulares:*** Android M trae mejoras en el apartado de privacidad. De tal manera que ahora los usuarios podemos decidir qué permisos otorgamos a las aplicaciones. Así podríamos decidir si queremos que una aplicación tenga acceso a nuestras imágenes, vídeos, llamadas, etc. Aterrizamos en estos controles desde el menú de ajustes, donde cada aplicación tendría listadas todas las funciones a las que potencialmente podría acceder para poder llevar a cabo todas sus acciones. En el ejemplo de abajo vemos lo que ocurre cuando accedemos a la aplicación de Facebook desde los ajustes. El apartado 'Permisos' nos permite darle acceso a la app a distintas funciones de nuestro teléfono.

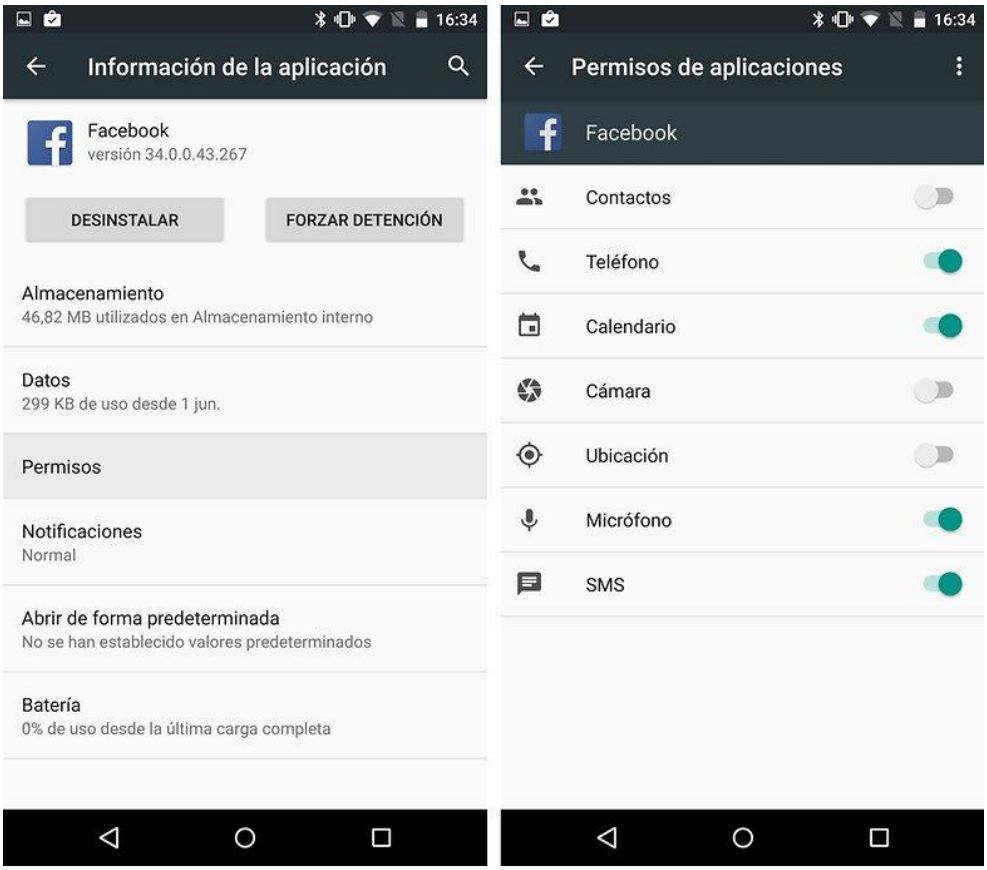


FIGURA 6: CONTROL DE PERMISOS GRANULARES

c) ***Chrome Custom Tabs:*** A través de esta función los desarrolladores pueden incrustar Chrome en su aplicación de tal manera que parece como un navegador dentro de la aplicación, pero es Chrome. Esta nueva función puede guardar contraseñas y está disponible desde hoy mismo para desarrolladores.

d) ***Personalización de los parámetros rápidos:*** El sistema UI tuner ahora permite editar desde el menú los parámetros rápidos que aparecen en la barra de acciones del teléfono. Desde el menú de 'Ajustes' se puede cambiar el orden en el que aparecen estos parámetros.

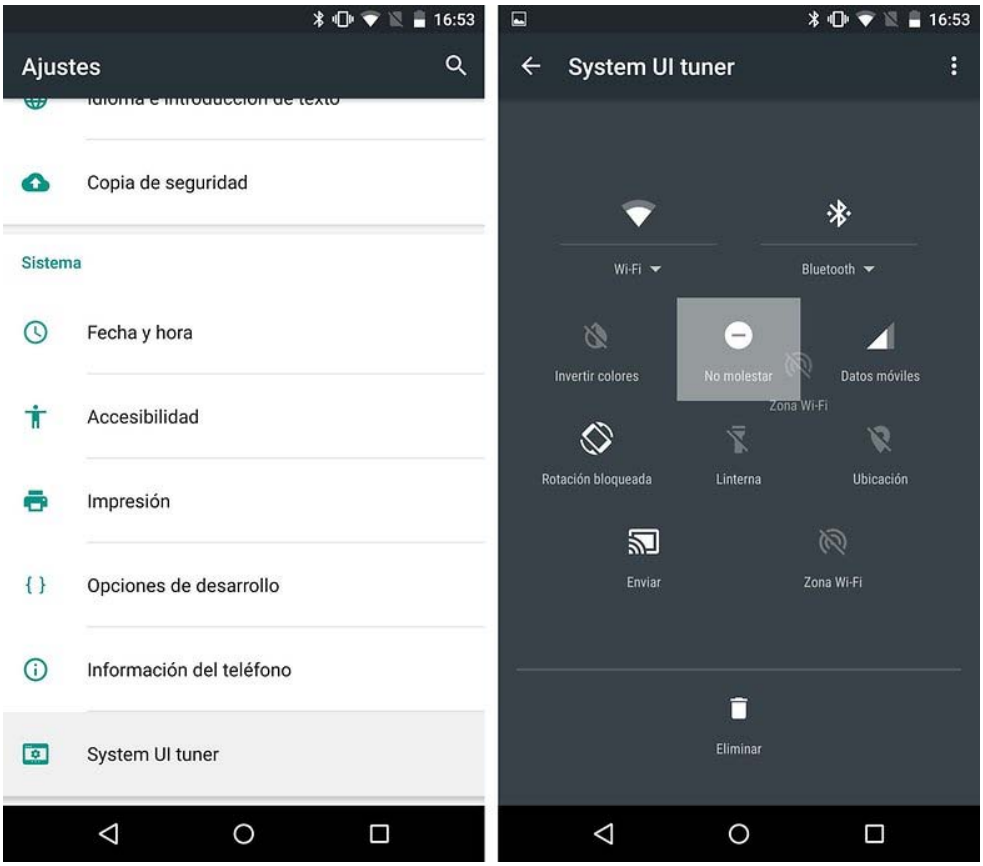


FIGURA 7: PERSONALIZACIÓN DE PARÁMETROS RÁPIDOS

e) **Android Pay:** Basado en NFC y Host Card Emulation, este sistema de pagos se basa en la "sencillez y seguridad". Android Pay es una plataforma abierta, por lo que podremos utilizar Google o cualquier otra aplicación como, por ejemplo, la de nuestro banco.

Google nos asegura que la "seguridad es lo más importante en Android Pay", de tal manera que utilizarán una cuenta virtual que representará la información de nuestra cuenta, pero no enviará nuestro crédito actual con el pago. Por último, si nos roban el teléfono podremos utilizar Android Device Manager para bloquearlo inmediatamente desde cualquier lugar.

f) **Desbloqueo por huella dactilar:** Google ha incluido una nueva autenticación por huella dactilar en Android M. Algunos terminales Android, como el Samsung Galaxy S6, ya incluyen esta función, pero a través del software del fabricante, no de Android, por lo que ahora Gogole ofrece soporte oficial. Esto significaría que el próximo Nexus también traería el hardware necesario para identificar la huella dactilar.

g) **Doze, mejora del uso de la batería:** Los esfuerzos de Google por mejorar la autonomía de la batería siguen en aumento. Con Android M se quiere mejorar el control de posicionamiento (que como sabemos consume la batería muy rápidamente), optimización del uso de la memoria RAM (que en algunos dispositivos es bastante alto), así como una reducción en el consumo cuando el terminal tiene la pantalla apagada o si no detecta ningún movimiento, haciendo que se desactiven algunas aplicaciones. Por ejemplo, un Nexus 9 con 'Doze' puede durar dos veces más que un Nexus 9 sin este nuevo sistema. Además, cabe destacar que Android soportará USB Type C y, por tanto, podremos colocar el cable sin prestar atención a la posición de la clavija.

FIGURA 8: CABLE USB TYPE C

h) **Otras mejoras:** Muchos usuarios se alegrarán al escuchar que con Android M se ha mejorado la selección de palabras, así como la forma de copiar y pegar. Además, también han traído de vuelta el modo silencio.

5.1.5 ¿Cuál será el nombre de Android M?

Al principio de la presentación de Keynote podíamos ver imágenes de la Vía Láctea, en inglés Milky Way. Pero algo todavía más premonitorio es que durante toda la presentación de Android M, el ponenete David Burke tuvo en la pantalla de su smartwatch un dibujo de un batido, Milkshake, en inglés. ¿Será ese su nombre?

Por supuesto la comunidad de internet se ha revolucionado al darse cuenta de este detalle. El protagonista ha salido al paso para quitarle importancia al asunto desde su cuenta de Twitter, declarando que el rumor es exagerado y afirmando que su reloj le muestra aleatoriamente sus postres favoritos.

Por el momento todo son rumores y no se ha confirmado nada. Desde Android Police se filtraba también un rumor que afirma que el nombre interno con el que Google está nombrando la nueva versión es MNC, Macadamia Nut Cookie. Dicha galleta también la podemos ver en las imágenes que David Burke ha colgado en su Twitter.

Como siempre, se trata de un dulce o de un postre, pero la apuesta todavía no está cerrada, así que hay más nombres con los que se especula de momento, como: Marshmallow, Muffin, Milky Bar, M&Ms, Macaroon y Mars.

FIGURA 9: POSIBLE NOMBRE DE ANDROID M

5.2 ¿Podrá ser Dart y no Java el Futuro de Android?

Dart para Android está diseñado para ser rápido, eficiente e integrado totalmente con los servicios web.

Las aplicaciones para Android están normalmente escritas en Java, pero uno de los numerosos grupos que trabajan en diferentes proyectos en Google está experimentando con una nueva forma de desarrollar para Android usando Dart, el lenguaje web diseñado por Google. El objetivo es escribir aplicaciones sin depender de Java y que sean más rápidas y con una mayor facilidad de estar integradas con el mundo web.

Dart está creado por los desarrolladores del popular Chrome V8 engine (Node.js), tras la frustración constante sufrida al desarrollar aplicaciones web con lenguajes no diseñados para ese menester debido a su antigüedad. En una conferencia sobre Dart se ha enseñado los primeros pasos de Dart para Android. Se llama Sky y es opensource. Está en una fase preliminar, pero es prometedor.

FIGURA 10: LOGO DE DART

El mayor objetivo del proyecto es brindar una nueva cuota de fluidez y rapidez a Android. Ahora mismo 60 fps (imágenes por segundo) es la tasa a la que los desarrolladores intentan llegar, aunque por desgracia no siempre se llegue a ello. El proyecto Sky establece una tasa de 120 fps que, por el momento, carece de sentido por los 60hz de refresco de los LCD y AMOLED que usan nuestros smartphones. Pero ya hay monitores de 120hz e incluso de 144hz con grandes mejoras a la fluidez con la que vemos el movimiento a través de la pantalla. A más refresco menos blur y screen tearing.

En la demo presentada en la conferencia, el hardware conseguía renderizar frames cada 1.2ms. Y aunque sea una demo, es bastante espectacular considerando que se tiene que renderizar una imagen cada 8ms para conseguir una tasa de 120 fps. El "truco" está en que

el hilo de la UI principal nunca se bloquea al llamar a APIs del sistema operativo, por lo que el usuario, aunque haya esperas y ralentizaciones, nunca las apreciará en la interfaz.

El otro punto clave de Dart es la web. Por ello no depende de ninguna plataforma en concreto y podrá ejecutarse la aplicación en Android, iOS o cualquier SO que tenga una máquina virtual Dart. Casi todas las llamadas ejecutadas en el código de la aplicación son servidas a través de HTTP, siendo las URL la base de Sky (Dart para Android). El inconveniente es que la aplicación no podrá funcionar cuando estás offline y la aplicación tardará 1 o 2 segundos en cargar debido a que tiene que ejecutar las llamadas HTTP y descargar la información, algo que se podría intentar arreglar con la caché.

Depender de llamadas HTTP tiene una gran ventaja: todo depende del servidor. El desarrollador no necesitará editar el código, compilarlo y mandar a revisión. El código estaría alojado en el servidor web y se serviría a todos los clientes, tan sólo tendrían que cerrar y abrir la aplicación para actualizarla. Es muy parecido a cómo funcionaría una página web.

En Android ya hay disponible un pequeño framework que permite diseñar la aplicación de forma sencilla con numerosos elementos Material Design. Las aplicaciones desarrolladas con Sky tendrían acceso a todas las API de Android, como una aplicación común desarrollada en Java, pero con la ventaja de estar integrada al 100% con la web y ser mucho más rápida y eficiente.

Depender de la web para ejecutar una aplicación es todavía una locura, pero con cada vez más usuarios conectados las 24 horas del día y las facilidades que supondría a los desarrolladores, no es descabellado que el futuro se asemeje a estas pinceladas en el presente.

6 ANDROID STUDIO

Android Studio es un entorno de desarrollo integrado (IDE) para la plataforma Android. Es el actual IDE oficial. Fue anunciado por Ellie Powers el 16 de Mayo de 2013 y, desde entonces, está disponible gratuitamente para desarrolladores. Está basado en IntelliJ IDEA de JetBrains, y diseñado específicamente para desarrollar para Android. Se puede descargar para Windows, Mac OS X y Linux.

FIGURA 11: LOGO DE ANDROID STUDIO

Características:

- Renderización en tiempo real
- Consola de Desarrollador: consejos de optimización, ayuda para la traducción, estadísticas de uso.
- Soporte para construcción basada en Gradle
- Refactorización específica de Android y arreglos rápidos
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones, y otros problemas.
- Plantillas para crear diseños comunes de Android y otros componentes.
- Soporte para programar aplicaciones para Android Wear.

6.1 Estructura de un Proyecto en Android Studio

El proyecto de una aplicación está contenido en una jerarquía donde se ubican todos los archivos de código fuente Java, los recursos, las configuraciones y los archivos de construcción de Gradle.

A continuación vamos a explicar cómo se estructura un proyecto Android dentro de Android Studio. También expondremos cómo está estructurado el archivo AndroidManifest.xml, que contiene la carpeta "res", y se verá la utilidad de los archivos de diseño y los archivos R.java.

6.1.1 Directorios de un Proyecto en Android Studio

Para comenzar la explicación, ejecutamos Android Studio y seleccionamos el proyecto "TweetMiw" que estamos desarrollando. En el panel de la izquierda, en la pestaña llamada "Project" es donde se representa la estructura del proyecto. Dentro de la carpeta principal del proyecto, se encuentran las carpetas de .gradle, .idea, app, build, gradle y otros archivos de configuración.

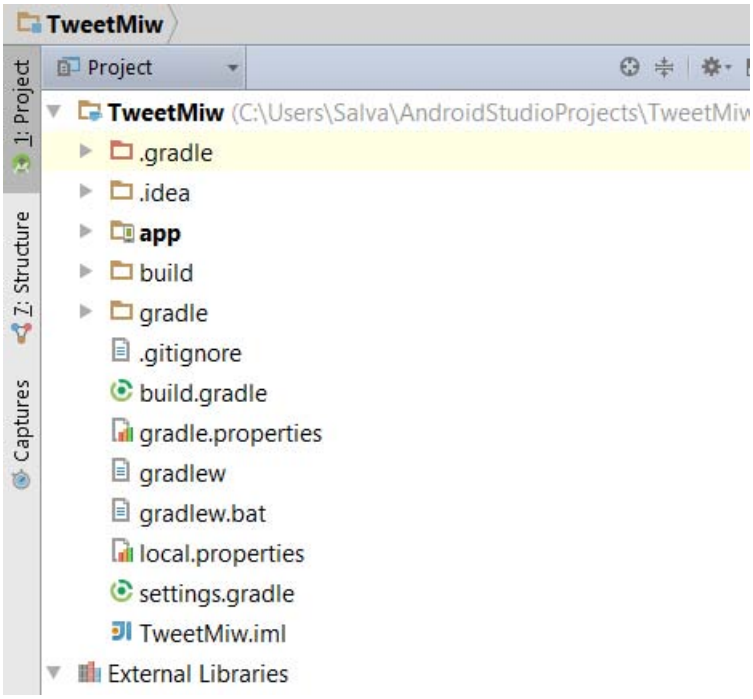


FIGURA 12: ESTRUCTURA DE UN PROYECTO EN ANDROID STUDIO

La carpeta app es la que contiene todo lo relacionado con el proyecto, es la carpeta en la que trabajamos y donde incluiremos los archivos necesarios para que nuestra aplicación sea empaquetada. Desplegando su contenido vemos dos carpetas: build y src.

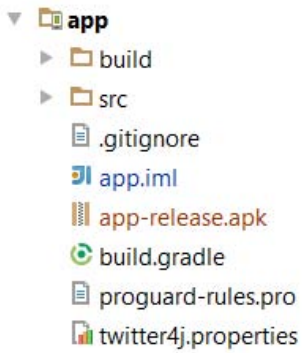


FIGURA 13: ESTRUCTURA DE LA CARPETA APP

La mayor parte del tiempo hemos estado trabajando sobre la carpeta src (Source). Dentro de ella se ubica la carpeta main, que a su vez contiene una carpeta java, donde se encuentran todos los archivos fuente Java para nuestra aplicación, una carpeta res (Resources) que contiene los recursos del proyecto (iconos, sonido, diseños, etc.) y el archivo AndroidManifest.xml.

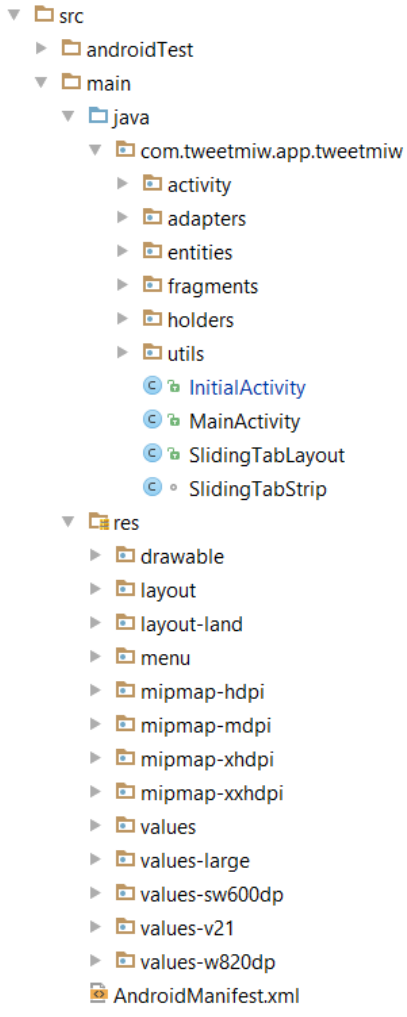


FIGURA 14: ESTRUCTURA DE LA CARPETA SRC (SOURCE)

6.1.2 ¿Qué es el Android Manifest?

Es un archivo XML que contiene nodos descriptivos sobre las características de una aplicación Android. Características como los building blocks existentes, la versión de SDK usada, los permisos necesarios para ejecutar algunos servicios y muchas más. En pocas palabras, el Android Manifest es el archivo general de configuración de toda nuestra aplicación.

Por ejemplo nuestro archivo de configuración es:

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tweetmiw.app.tweetmiw" >

    <uses-permission android:name="android.permission.INTERNET" />
    <!-- Network State Permissions -->
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".InitialActivity"
            android:label="@string/app_name" >
        </activity>
        <activity
            android:name=".activity.TweetDetailActivity"
            android:label="@string/app_name" >
        </activity>
        <activity
            android:name=".activity.WriteTweetActivity"
            android:label="@string/app_name" >
        </activity>
        <meta-data
            android:name="io.fabric.ApiKey"
            android:value="535e40f880186e87c78e2a6a07186424d8232012" />
    </application>

</manifest>
```

FIGURA 15: ARCHIVO ANDROID MANIFEST

Todas las aplicaciones deben contener este archivo por convención. El nombre debe permanecer intacto, ya que se usa como referencia para el parsing de nuestra aplicación. El nodo raíz de este documento se representa con la etiqueta <manifest> y por obligación debe contener un hijo de tipo <application>.

En el código anterior podemos observar que manifest posee dos atributos, xmlsn:android y package. El primero no debemos cambiarlo nunca, ya que es el namespace del archivo.

El atributo package indica el nombre del paquete Java que soporta a nuestra aplicación. El nombre del paquete debe ser único y un diferenciador a largo plazo.

La etiqueta <application> representa cómo estará construida nuestra aplicación. Dentro de ella definiremos nodos referentes a las actividades que contiene, las librerías incluidas, los Intents, Providers, y demás componentes.

Algunos atributos de la etiqueta <application> son:

- allowBackup: Este atributo puede tomar los valores de true o false. Indica si la aplicación será persistente al cerrar nuestro AVD.
- icon: Indica dónde está ubicado el icono que se usará en la aplicación. Debemos indicar la ruta y el nombre del archivo que representa el icono. En este caso apuntamos a la carpeta mipmap donde se encuentra ic_launcher.png.
- label: Es el nombre de la aplicación que verá el usuario en su teléfono. Normalmente apunta a la cadena "app_name" que se encuentra en el recurso strings.xml.
- theme: Este atributo apunta al archivo de recursos styles.xml, donde se define la personalización del estilo visual de nuestra aplicación.

Dentro de <application> encontraremos expresada la actividad principal que definimos al crear el proyecto. Usaremos <activity> para representar un nodo tipo actividad.

Dentro de <activity> hay dos atributos: name, el cual se refiere a la clase Java que hace referencia a esta actividad (comienza con un punto "."); y el atributo label que hace referencia al texto que se mostrará en la cabecera de la actividad. En este caso es el mismo string "app_name".

6.1.3 ¿Qué utilidad tiene el archivo strings.xml?

Dentro de la carpeta "res" encontraremos todos aquellos recursos necesarios para nuestra aplicación. Esta práctica de excluir los atributos de la aplicación a través de archivos externos, permite reducir la complejidad de diseño en las interfaces.

Uno de los recursos más relevantes es el archivo strings.xml que se encuentra dentro de la subcarpeta values. Este fichero almacena todas las cadenas que se muestran en los widgets (controles, formas, botones, vistas, etc) de nuestras actividades.

Por ejemplo, si tenemos un botón cuyo título es "Presiona aquí", es recomendable incluir dicha cadena en el archivo strings.xml.

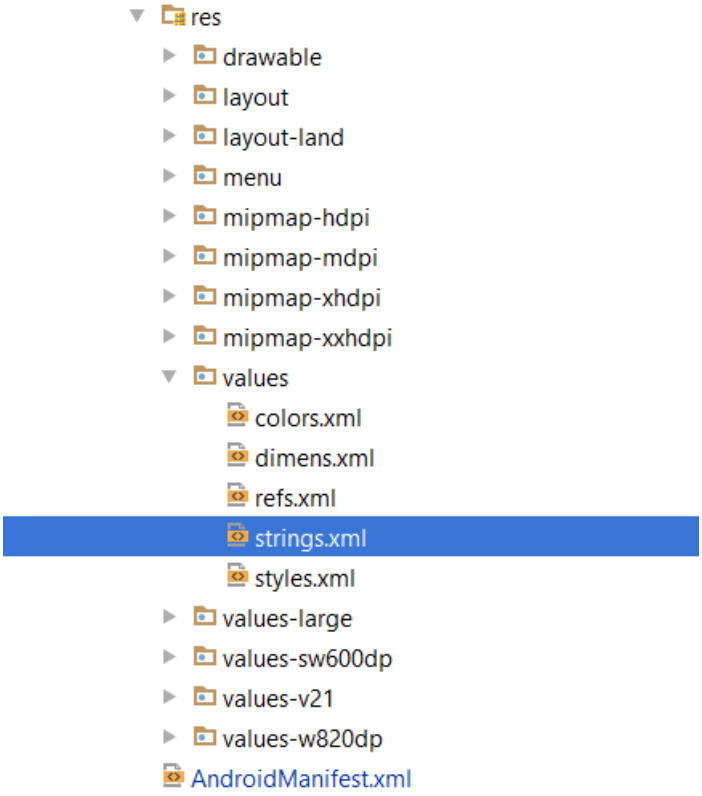


FIGURA 16: RUTA DEL ARCHIVO STRINGS

Para declarar nuestro recurso de strings usaremos el nodo raíz <resources>. Para declarar las cadenas usaremos la etiqueta <string> y estableceremos el atributo name como identificador. Dentro de esta etiqueta pondremos el texto que se visualizará en el componente de interfaz.

Dentro de este archivo nos encontramos con todos las cadenas de texto que utilizamos en nuestra aplicación.

strings.xml ×

```
<resources>
    <string name="app_name">TweetMiw</string>

    <string name="action_settings">Configuración</string>
    <string name="login">Login</string>
    <string name="title_activity_initial">InitialActivity</string>
    <string name="tweets_pendientes">Tweets Pendientes</string>
    <string name="cerrar_sesion">Cerrar Sesión</string>
    <string name="ayuda">¿Necesitas ayuda?</string>
    <string name="lista">lista</string>
    <string name="timeline">Timeline</string>
    <string name="search">Buscar</string>


    <string name="description_close">Botón de cerrar</string>
    <string name="description_photo">Botón para insertar una foto</string>
    <string name="description_responder">Botón para responder un tuit</string>
    <string name="description_retuitear">Botón para retuitear</string>
    <string name="description_favorito">Botón para marcar el tuit como favorito</string>
    <string name="description_avatar">Imagen del avatar del usuario</string>
    <string name="description_seguir">Imagen para seguir a otro usuario</string>
    <string name="description_logo">Imagen del logo de la aplicación</string>


    <string name="msg_yes">Si</string>
    <string name="msg_no">No</string>
    <string name="msg_recommendation">¿Nos recomendarías?</string>
    <string name="msg_front_desk">¿cómo consideras el servicio de la recepción?</string>
    <string name="msg_front_desk_0">Malo</string>
    <string name="msg_front_desk_100">Bueno</string>
    <string name="dialog_title">¿Desea enviar los datos?</string>
    <string name="title_tab_list">Listado</string>
    <string name="title_tab_grid">Vista</string>
    <string name="title_activity_main">Principal</string>
    <string name="list_separator_standard">Estándar</string>
    <string name="list_separator_luxury">De lujo</string>
    <string name="twittear">Twittear</string>
    <string name="hint_twittear">¿Qué está pasando?</string>


<!-- TODO: Remove or change this placeholder text -->
    <string name="hello_blank_fragment">Hello blank fragment</string>
    <string name="user_loading">cargando</string>
    <string name="label_tweets_not_found">Tuits no encontrados. Compruebe su conexión a internet.</string>
    <string name="label_tweets_downloaded">Tuits descargados</string>
    <string name="label_login_noAuth">Login No Autorizado</string>
    <string name="label_tweet_published">Tweet publicado</string>
    <string name="label_tweet_Nopublished">Error publicando el Tweet. Compruebe su coneción a internet.</string>
    <string name="label_tweet_followers">Siguiendo</string>
    <string name="label_tweet_following">Seguidores</string>
</resources>
```

FIGURA 17: ARCHIVO STRINGS

El archivo strings.xml es muy útil para los desarrolladores. Una de sus grandes utilidades es facilitar el uso de múltiples idiomas en tu aplicación. Esto se debe a que puedes externalizar las cadenas del código java y seleccionar la versión del archivo strings.xml con el lenguaje necesitado.

6.1.4 ¿Qué hay dentro de la carpeta layout?

En la carpeta layout encontramos los archivos de diseño de todas las actividades. En nuestro caso en esta carpeta también hemos incluido los fragmentos que hay dentro de esas actividades.

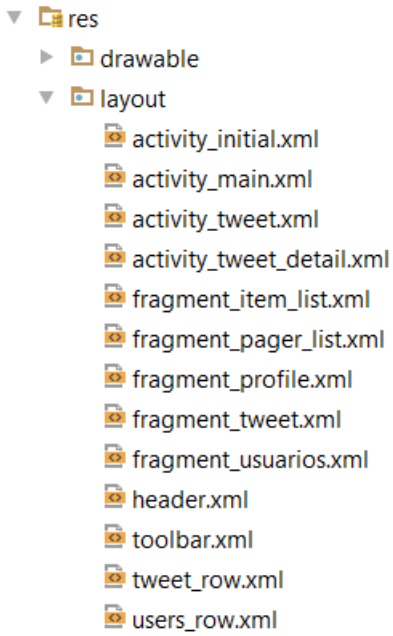


FIGURA 18: ESTRUCTURA DE LA CARPETA LAYOUT

Construir la interfaz a través de nodos XML es mucho más sencillo que la creación a través de código Java. Adicionalmente Android Studio nos ha dotado de un panel de diseño estilo Drag and Drop, lo cual facilita la creación de una interfaz de usuario por parte de los desarrolladores.

Las actividades comienzan con un nodo raíz llamado <RelativeLayout>. Un Layout es el contenedor principal que define el orden y secuencia en que se organizarán los widgets en nuestra actividad. Existen varios tipos de Layouts, como por ejemplo el RelativeLayout, LinearLayout, GridLayout, FrameLayout, etc.

Android Studio crea por defecto un RelativeLayout porque permite crear un grupo de componentes con ubicaciones relativas. Quiere decir que se ubicarán por referencias y no por valores absolutos. Esto permite ajustar nuestras aplicaciones a cualquier tipo de pantalla para dispositivos móviles.

Un ejemplo de actividad sería el siguiente, que representa al archivo activity_main.xml:

activity_main.xml ×

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity"
    android:background="@color/primary_color">

    <com.twitter.sdk.android.core.identity.TwitterLoginButton
        android:id="@+id/login_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:src="@mipmap/logo"
        android:contentDescription="Imagen del logo de la aplicación"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="109dp" />


    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="¿Necesitas ayuda?"
        android:textSize="18sp"
        android:textColor="@color/tw__solid_white"
        android:id="@+id/button3"
        android:layout_below="@+id/login_button"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="48dp"
        android:onClick="alertDialog" />


</RelativeLayout>
```

FIGURA 19: ARCHIVO ACTIVITY_MAIN

Veamos la utilidad de los atributos para <RelativeLayout>:

- layout_width: Es el ancho que tendrá el layout dentro de la actividad. Aunque se puede especificar con unidades personalizadas, es recomendable usar match_parent para ajustarlo al ancho del dispositivo.
- layout_height: Representa la dimensión vertical del layout. Usa match_parent para ajustarla al dispositivo.
- paddingLeft, paddingRight: Es el espacio lateral existente entre el contorno del Layout y los widgets. Su valor puede estar definido directamente o apuntar al archivo

de recursos dimens.xml ubicado en la carpeta "values". Este archivo contiene nodos de tipo <dimen> con valores density-indepent pixels (dp).

- paddingTop, paddingBottom: Es el espacio vertical existente entre el contorno del Layout y los widgets
- context: Define el nombre del archivo Java que contiene la actividad donde el Layout será acogido.
- background: Define el color de fondo que tiene esa actividad.

6.1.5 La carpeta mipmap en Android Studio

Todas las imágenes de la aplicación se encuentran ubicadas en las carpetas que comienzan por "mipmap". Estas carpetas se relacionan directamente con el tipo de densidad de pantalla donde se ejecutará nuestra aplicación.

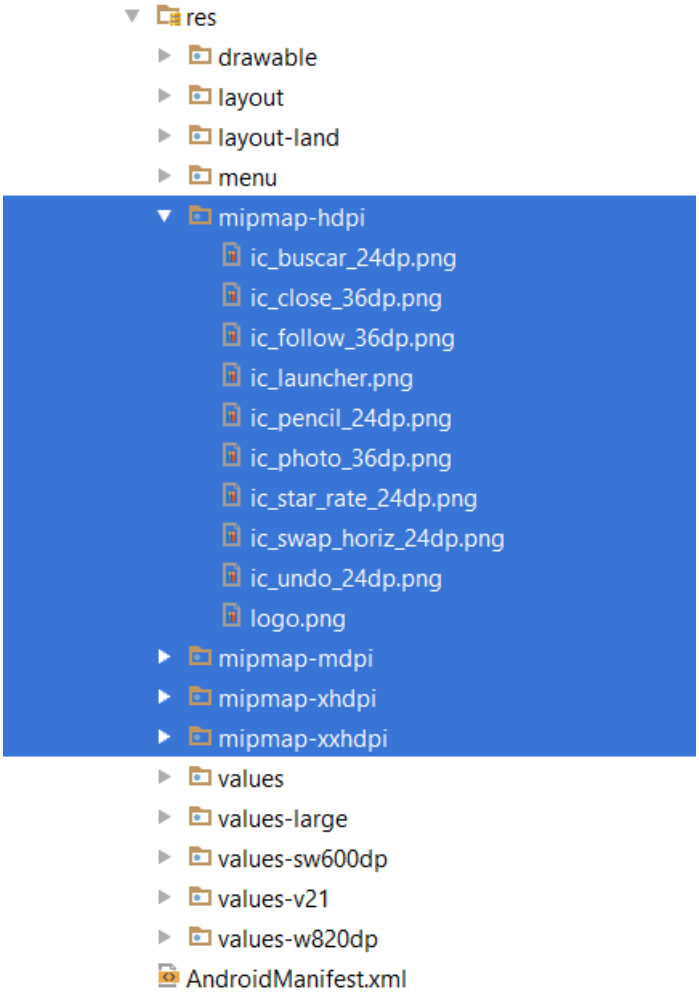


FIGURA 20: CONTENIDO DE LA CARPETA MIPMAP

La mayoría de dispositivos móviles actuales tienen uno de estos 4 tipos de densidades:

- Mediun Dots per Inch (mdpi): Este tipo de pantallas tienen una densidad de 160 puntos por pulgada.
- High Dots per Inch (hdpi): En esta clasificación encontraremos teléfonos cuya resolución es de 240 puntos por pulgada.
- Extra high dots per inch (xhdpi): Resoluciones mayores a 340 puntos por pulgada.
- Extra Extra high dots per inch (xxhdpi): Rangos de resoluciones mayores a 480 puntos por pulgada.

Miremos una pequeña imagen ilustrativa:

FIGURA 21: DIFERENTES DENSIDADES DE PANTALLA

La imagen muestra algunos ejemplos de dispositivos móviles cuya densidad se encuentra dentro de los rangos establecidos. Al final podemos ver una categoría extra

llamada xxxhdpi. Esta denominación describe a la densidad de televisores de última generación que ejecutan Android.

En cada una de las carpetas hemos definido los recursos gráficos en los distintos tipos de densidades. Esto nos permite tener mayor compatibilidad en la aplicación con distintos dispositivos móviles.

6.1.6 Comprendiendo el propósito de la clase R.java

El archivo R.java es una archivo que se autogenera dentro de la carpeta build, para linkear todos los recursos que tenemos en nuestro proyecto al código Java.

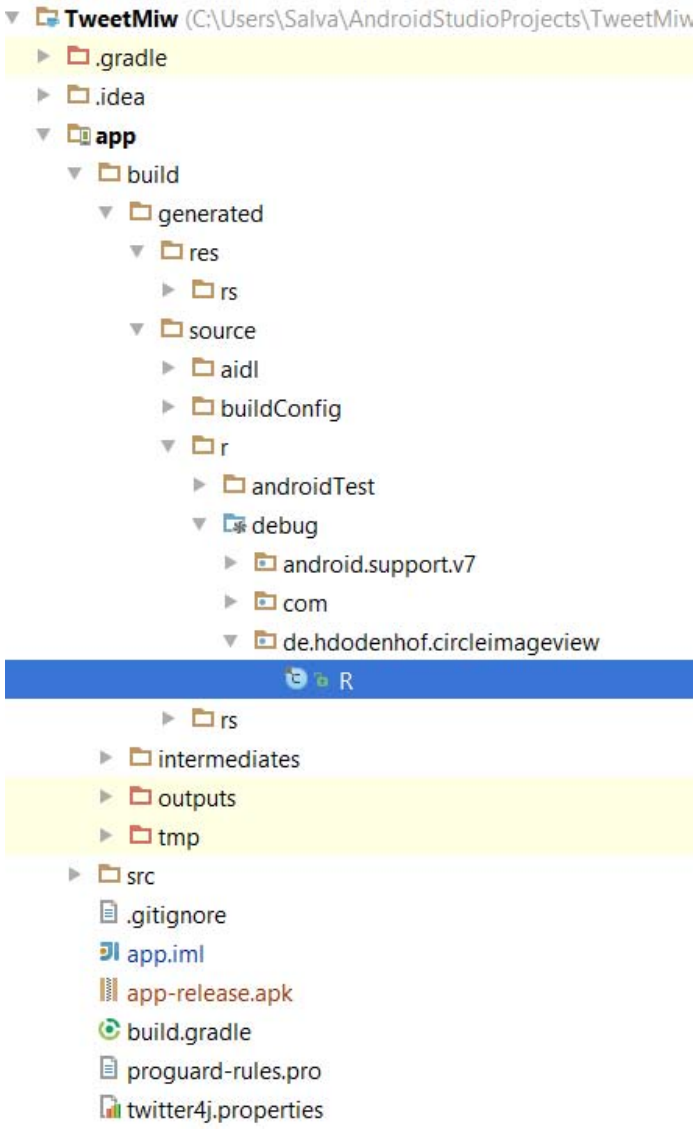


FIGURA 22: LOCALIZACIÓN DE LA CLASE R DE JAVA

Si abres el archivo podremos ver un código similar a este:

appcompat\R.java ×

Files under the build folder are generated and should not be edited.

```
/.../
package android.support.v7.appcompat;

public final class R {
    public static final class anim {
        public static final int abc_fade_in = 0x7f050000;
        public static final int abc_fade_out = 0x7f050001;
        public static final int abc_grow_fade_in_from_bottom = 0x7f050002;
        public static final int abc_popup_enter = 0x7f050003;
        public static final int abc_popup_exit = 0x7f050004;
        public static final int abc_shrink_fade_out_from_bottom = 0x7f050005;
        public static final int abc_slide_in_bottom = 0x7f050006;
        public static final int abc_slide_in_top = 0x7f050007;
        public static final int abc_slide_out_bottom = 0x7f050008;
        public static final int abc_slide_out_top = 0x7f050009;
    }
    public static final class attr {
        public static final int actionBarDivider = 0x7f010083;
        public static final int actionBarItemBackground = 0x7f010084;
        public static final int actionBarPopupTheme = 0x7f01007d;
        public static final int actionBarSize = 0x7f010082;
        public static final int actionBarSplitStyle = 0x7f01007f;
        public static final int actionBarStyle = 0x7f01007e;
        public static final int actionBarTabBarStyle = 0x7f010079;
        public static final int actionBarTabStyle = 0x7f010078;
        public static final int actionBarTabTextStyle = 0x7f01007a;
        public static final int actionBarTheme = 0x7f010080;
        public static final int actionBarWidgetTheme = 0x7f010081;
        public static final int actionButtonStyle = 0x7f01009d;
        public static final int actionDropDownStyle = 0x7f010099;
        public static final int actionLayout = 0x7f010048;
        public static final int actionMenuTextAppearance = 0x7f010085;
        public static final int actionMenuTextColor = 0x7f010086;
        public static final int actionModeBackground = 0x7f010089;
        public static final int actionModeCloseButtonStyle = 0x7f010088;
        public static final int actionModeCloseDrawable = 0x7f01008b;
        public static final int actionModeCopyDrawable = 0x7f01008d;
        public static final int actionModeCutDrawable = 0x7f01008c;
        public static final int actionModeFindDrawable = 0x7f010091;
        public static final int actionModePasteDrawable = 0x7f01008e;
        public static final int actionModePopupWindowStyle = 0x7f010093;
        public static final int actionModeSelectAllDrawable = 0x7f01008f;
        public static final int actionModeShareDrawable = 0x7f010090;
        public static final int actionModeSplitBackground = 0x7f01008a;
```

FIGURA 23: ARCHIVO R DE JAVA

La clase R contiene clases anidadas que representan todos los recursos de nuestro proyecto. Cada atributo tiene una dirección de memoria asociada referenciada a un recurso en específico.

Nunca se debe modificar el archivo R.java, él se actualiza automáticamente al añadir un nuevo elemento al proyecto.

6.1.7 La carpeta java de un proyecto en Android Studio

Finalmente llegamos a la carpeta "java". Aquí se alojarán todos los archivos relacionados con nuestras actividades y otros archivos fuente auxiliares.

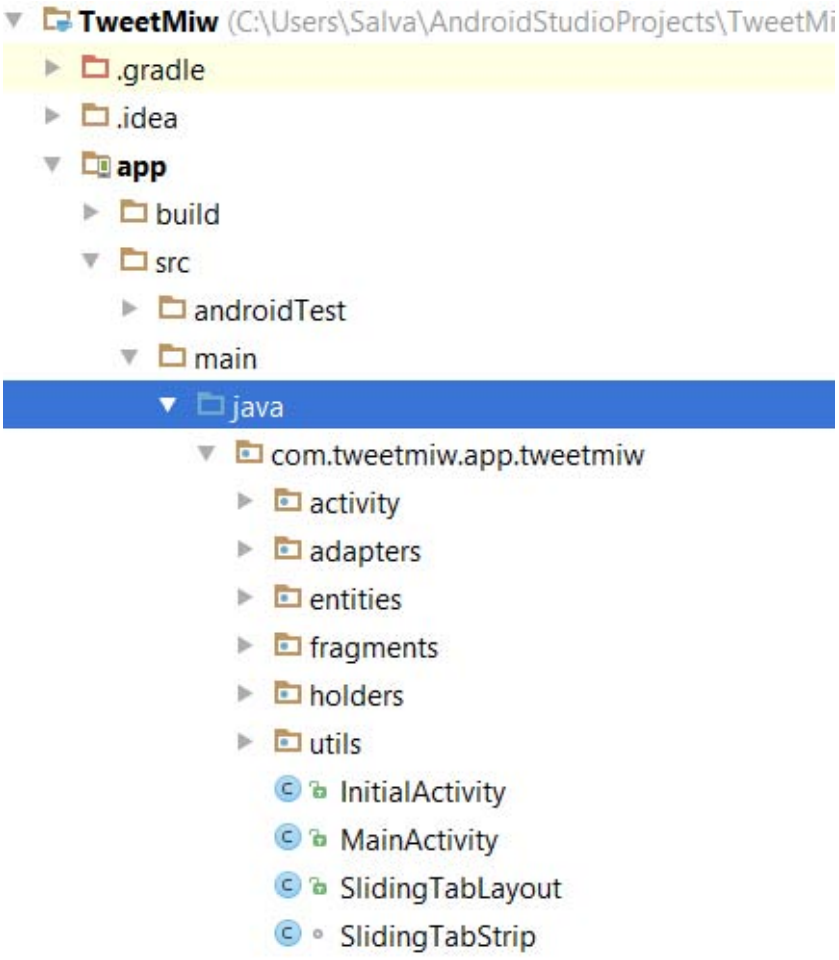


FIGURA 24: ESTRUCTURA DE LA CARPETA JAVA

Por poner un ejemplo, mostraremos el archivo TweetDetailActivity.java, donde vemos toda la lógica necesaria para que la actividad interactúe de manera correcta con el usuario.

TweetDetailActivity.java ×

```
package com.tweetmiw.app.tweetmiw.activity;

import ...

public class TweetDetailActivity extends AppCompatActivity {
    Tweet tweet;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tweet_detail);
        if (getResources().getConfiguration().orientation == Configuration.ORIENTATION_LANDSCAPE) {
            // If the screen is now in landscape mode, we can show the
            // dialog in-line with the list so we don't need this activity.
            finish();
            return;
        }

        if (savedInstanceState == null) {
            // During initial setup, plug in the details fragment.
            TweetDetailFragment details = new TweetDetailFragment();
            details.setArguments(getIntent().getExtras());

            //getFragmentManager().beginTransaction().add(R.id.contentPanel,details) add(android.R.id.content, details).commit();

        }
        Toolbar toolbar = (Toolbar) findViewById(R.id.activity_my_toolbar);
        setSupportActionBar(toolbar);//modifico el action Bar pordefecto de l
    }
}
```

FIGURA 25: ARCHIVO TWEETDETAILACTIVITY

Una actividad tiene un ciclo de vida y lo único que está bajo nuestro control es la manipulación de cada estado.

6.2 Estados de una Actividad Android

Estos son los estados de una actividad Android:

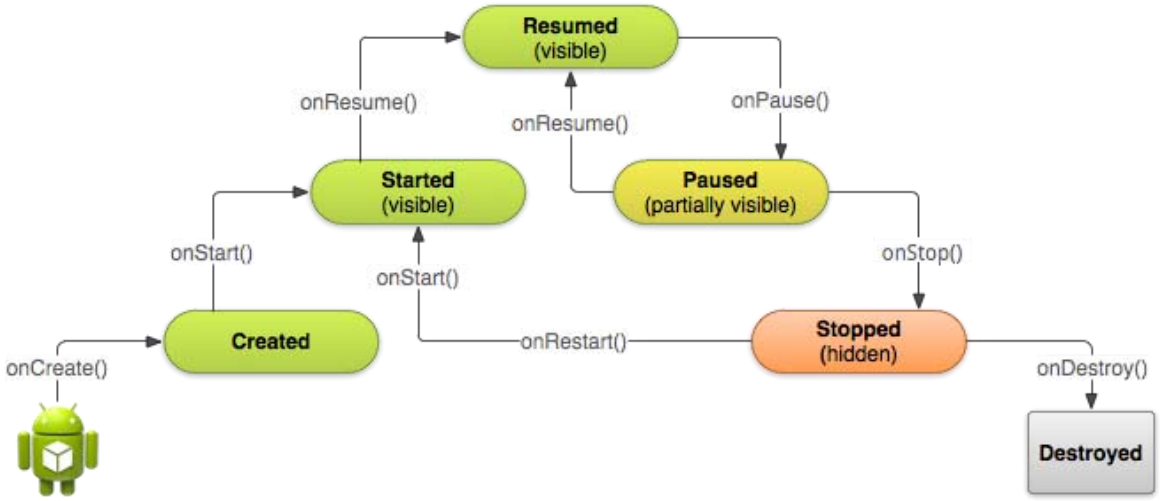


FIGURA 26: ESTADOS DE UNA ACTIVIDAD ANDROID

Toda actividad hereda de Activity y antes que eso de Context. Por ello, toda actividad debe sobreescribir el método *onCreate()*, que se ejecuta nada más iniciarse la actividad. Además, es éste el único método que estaremos obligados a implementar de todo el flujo de la actividad; los demás son opcionales. Después, la actividad se inicia, y se ejecuta el código que haya en *onStart()*. Estos dos estados son transicionales y se ejecutan rápidamente. A continuación, se ejecuta el código que haya en *onResume()*.

A partir de este momento, la actividad cambia entre los estados Resumed, Paused (cuando pierde el foco en el terminal del usuario, normalmente porque aparece un diálogo) y Stopped (cuando pasa a background porque se inicia otra actividad, posiblemente de otra aplicación diferente). Finalmente, una actividad pasa a estado Destroyed cuando finaliza, bien porque dicha actividad ha terminado automáticamente llamando a *finish()* o bien porque el usuario la cierra, pulsando sobre el botón de ir hacia atrás.

Cabe destacar que antes de pasar a Stopped, siempre se pasa antes por Paused, de manera que antes de ejecutar *onStopped()* se ejecutará *onPaused()*.

Los únicos estados estables en el tiempo son, por tanto, Resumed, Paused y Stopped. Cuando una actividad se inicia rápidamente se ejecuta el código de crear, iniciar y continuar (en inglés, resume). Finalmente, no es necesario sobrescribir *onDestroy()* a menos que haya que liberar recursos creados en las etapas de creación o inicialización.

A partir de ahí la actividad está plenamente operativa. Es importante tener esto en cuenta a la hora de diseñar las actividades y saber cuál será el flujo de ejecución de las mismas.

7 MATERIAL DESIGN

Material Design es un concepto, una filosofía, unas pautas enfocadas al diseño utilizado en Android, pero también en la web y en cualquier plataforma, desarrollado por Google y anunciado en la conferencia Google I/O celebrada el 25 de junio de 2014. Material Design consiste en un diseño limpio, en el que predominan animaciones y transiciones de respuesta, el relleno y los efectos de profundidad tales como la iluminación y las sombras.

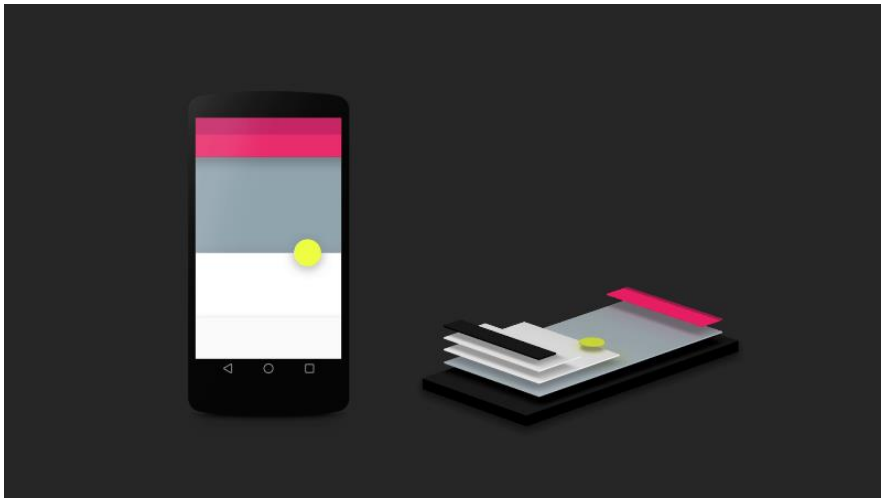


FIGURA 27: RELIEVE EN MATERIAL DESIGN

Material Design es utilizado en Android Lollipop, la nueva versión del sistema operativo Android que fue lanzada en otoño de 2014. Material Design también es usado en Google Drive y Google Docs, Sheets y Slides, y se irá extendiendo progresivamente a todos los productos de Google (incluyendo Google Search, Gmail y Google Calendar), proporcionando una experiencia consistente en todas las plataformas. Google también lanzó APIs para que los desarrolladores externos incorporaran Material Design a sus aplicaciones.

7.1 ¿En qué se basa Material Design?

Material Design recibe su nombre por estar basado en objetos materiales. Piezas colocadas en un espacio (lugar) y con un tiempo (movimiento) determinado.

Es un diseño donde la profundidad, las superficies, los bordes, las sombras y los colores juegan un papel principal.

Precisamente este diseño basado en objetos es una manera de intentar aproximarse a la realidad, algo que en un mundo donde todo es táctil y virtual es difícil. Material Design quiere guiarse por las leyes de la física, donde las animaciones sean lógicas, los objetos se superpongan pero no puedan atravesarse el uno al otro y demás.

¿Cómo se traslada esto a Android? Pues básicamente delimitando claramente el tipo de menús, los botones y los tipos de imágenes a elegir.

- ***Elementos ordenados e imágenes claras:*** Material Design es un diseño con una tipografía clara, casillas bien ordenadas, colores e imágenes llamativos para no perder el foco y un sentido del orden y la jerarquía muy marcado. Estas ideas ya se aplican en muchos diseños, pero en Material Design, Google ha creado unas normas muy claras de cómo llevarlo a la práctica.

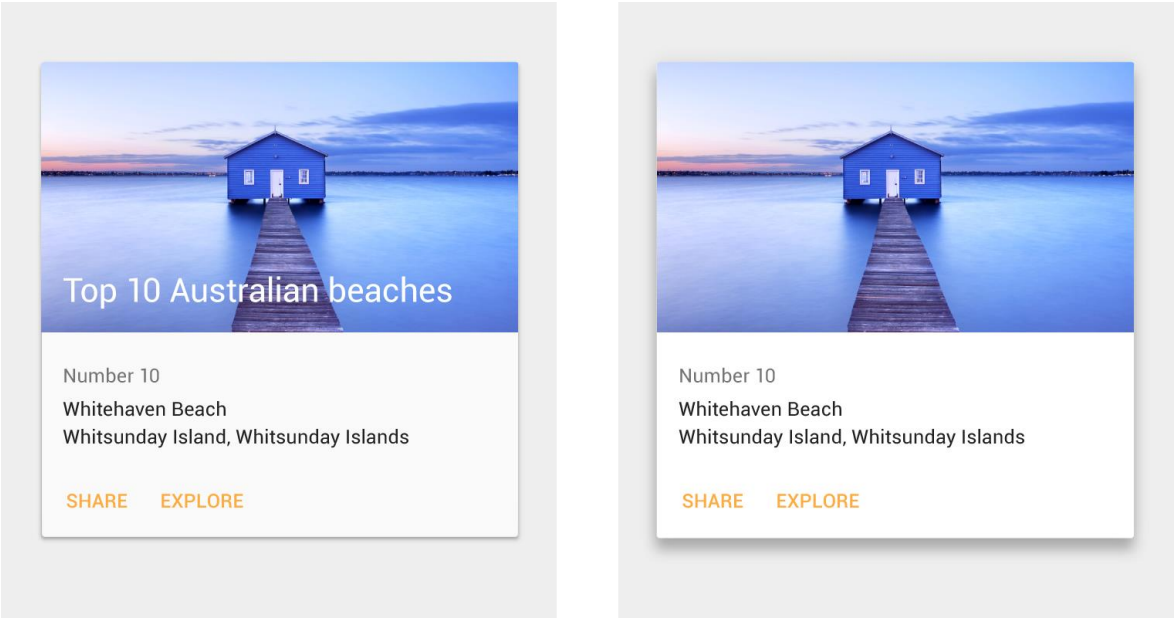


FIGURA 28: ELEMENTOS ORDENADOS

- ***Luz y sombras dan sensación de jerarquía:*** Uno de los elementos clave es la luz y las sombras. Una iluminación realista proporciona indicios de cómo se comportará un elemento y en qué nivel se encuentra. Imaginemos un cuadrado que tiene su sombra sobre otro cuadrado, el cerebro humano entiende que el que recibe la sombra es el que está debajo.

FIGURA 29: LUZ Y SOMBRAS EN ANDROID

- ***El movimiento es la mejor forma de guiar al usuario:*** El movimiento es otro elemento clave, por ejemplo un objeto que parpadea significa que está llamando tu atención**,** un elemento que se expande es que se acaba de abrir. Todos estos movimientos se crean en una dirección determinada, nuevamente explicada en la guía de Material Design.

- ***Animaciones, el elemento más característico:*** La velocidad en la que aparecen los elementos son fundamentales para dar esa sensación de realismo, por lo que los elementos no aparecen de repente. Finalmente está la dirección desde la que aparecen ya que con ese movimiento se está indicando al usuario de donde proviene la información. Sin duda las animaciones son uno de los aspectos más llamativos de Material Design.

7.2 Material Design, un Diseño Adaptado Para Todo Tipo de Pantallas

No debemos pensar en Material Design como ese diseño destinado para las aplicaciones móviles de Android. De hecho, es multiplataforma**.** Tantos los smartphones, tablets, smartwatches o televisores pueden hacer uso de este diseño. También las páginas webs. Material Design ha sido creado pensando en todos los sistemas, no solo Android.

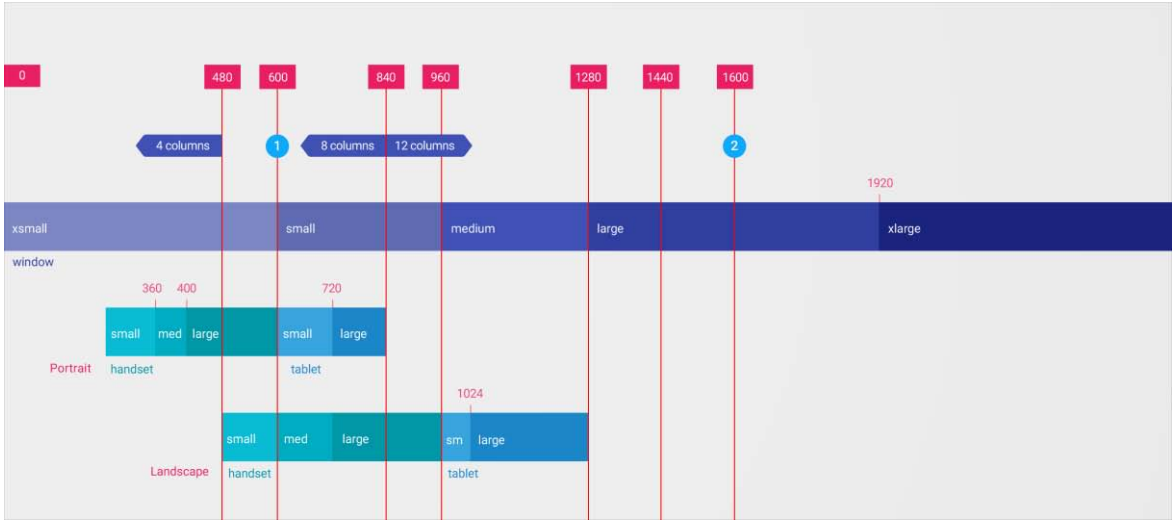


FIGURA 30: RESOLUCIONES DE PANTALLA

Material Design tiene sus propias normas para casi todos los detalles y se mantienen independientemente del tamaño de pantalla. Precisamente esa transversalidad es su punto fuerte.

Los usuarios de Android ya pueden disfrutar de una larga lista de aplicaciones actualizadas con algunos elementos de Material. Es verdad que todavía hay muchas incongruencias y fallos, y otras muchas sólo han incorporado detalles, pero los esquemas están claros y sólo es cuestión de tiempo que los desarrolladores sepan aplicar este diseño y adaptarlo a su aplicación.

Uno de los puntos más criticados de las apps con Material Design es que parece que todas sean iguales, pero esto es simplemente porque en esta fase inicial se están aplicando los modelos básicos.

Material Design es probablemente el cambio más importante de Android 5.0 Lollipop. Un cambio que no sólo afecta a Android, también define un poco la estrategia de Google.

7.3 Claves de Material Design que hemos Aplicado

En Material Design la selección de colores se limita a 3 tonalidades de la paleta de colores primaria y un color acentuado de la paleta de colores secundaria.

En nuestro caso, la selección de colores que hemos elegido ha sido:

FIGURA 31: COLORES SELECCIONADOS PARA LA APLICACIÓN

Google facilita una paleta de colores de donde lograr los colores idóneos para la aplicación.

También hemos tenido en cuenta a Material Design a la hora de elegir el Theme general de la aplicación. A pesar de que hay un nuevo Theme Material integrado en Android Studio, éste no es soportado por versiones anteriores a la API 21, por lo que hemos utilizado el AppCompat sin el action bar, que es la mejor opción para llevar Material Design a versiones anteriores.

Otro factor que hemos procurado tener en cuenta son los márgenes, ya que en esto Google también te orienta sobre cuál debe ser la distancia entre los objetos y los márgenes.

8 METODOLOGÍAS

8.1 Metodología de Gestión

En nuestro caso para realizar el proyecto hemos utilizado una metodología ligera como es Scrum. Nos hemos decidido por esta metodología puesto que da una mayor flexibilidad a la hora de trabajar y permite identificar las funcionalidades básicas que necesitas que realice la aplicación y luego poder añadirle funciones extra de una forma más dinámica, lo que en nuestro caso era fundamental por tener un tiempo muy limitado.

Otro de los motivos por los que nos hemos decidido por Scrum es porque en esta metodología de trabajo el equipo es "auto-organizado" (auto-gestionado), con un alto grado de autonomía y responsabilidad. Además siendo un equipo tan reducido (de sólo dos personas), necesitábamos auto-organizarnos ya que no podíamos ser dirigidos por un jefe de equipo o un jefe de proyecto.

Al ser nosotros mismos los dueños del producto, no necesitábamos esa vía de comunicación y, por ejemplo, no necesitábamos la figura del Scrum Master.

8.2 Metodología de Desarrollo

Como metodología de desarrollo, hemos utilizado Extreme Programming debido a que es la más indicada para equipos de trabajo pequeños o medianos (de 2 a 10 programadores) como era en nuestro caso.

Esto nos ha permitido hacer iteraciones cortas y estar constantemente revisando el código para simplificarlo. Además de eso, hemos podido repartirnos los sprints de forma que cada uno pudiese tomar las decisiones que considerara mejores, y no se tomaran aquellas para las que no fuéramos los mejor preparados.

9 ARQUITECTURA

La arquitectura de Android [4] está formada por los siguientes componentes: núcleo Linux, runtime de Android, bibliotecas, marco de trabajo de las aplicaciones y aplicaciones, como se puede ver en la figura.

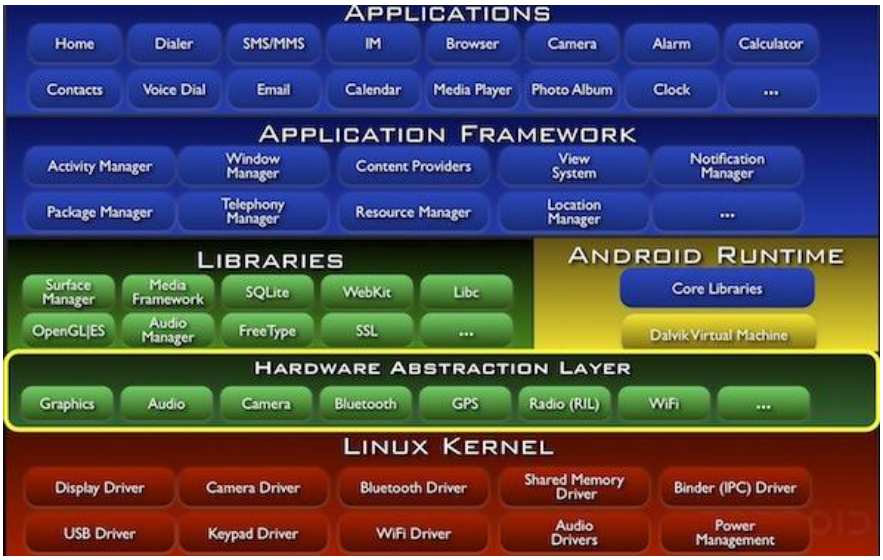


FIGURA 32

- ***El núcleo Linux:*** El núcleo de Android está basado en el sistema operativo Linux LTS (long term support) desde 2014, Android usa principalmente las versiones 3.4 ó 3.10 de Linux. La versión del núcleo de Android depende sobre todo del hardware con el que se ha desarrollado el dispositivo Android. Esta capa es la de abstracción entre el hardware y el resto de la pila, siendo la única dependiente del hardware.

- ***Runtime de Android:*** Hasta la versión 5.0 Android uso la máquina virtual de Dalvik [5] se basa en el concepto de máquina virtual utilizado en Java. Esta máquina ejecuta ficheros Dalvik ejecutables (.dex), basada en registros. Cada aplicación corre en su propio proceso Linux con su propia instancia de la máquina virtual Dalvik. Sin embargo, Android 4.4 introduce el uso de Android Runtime (ART), una máquina virtual proceso que lleva a cabo la transformación de código de bytes de la aplicación en instrucciones nativas que luego son ejecutados por el entorno de ejecución del dispositivo. Esta máquina virtual se incluyó en la versión 4.4 pero no es hasta la 5.0 que se

empezó a usar. En esta capa también se encuentra el "core libraries" con la mayoría de las librerías disponibles en el lenguaje Java.

- ***Librerías nativas:*** Incluye un conjunto de librerías en C/C++, compiladas en código nativo del procesador. Muchas de las librerías utilizan proyectos de código abierto.

- ***Entorno de aplicación:*** Aquí se encuentra la plataforma de desarrollo libre para aplicaciones con muchas innovaciones (sensores, localización, servicios, barra de notificaciones,). En este entorno es donde más se aprovecha el lenguaje de programación Java ya que el SDK de Android no acaba de ofrecer todo lo disponible para su estándar del entorno de ejecución Java (JRE).

- ***Aplicaciones:*** Este nivel está formado por el conjunto de aplicaciones instaladas en una máquina Android. Todas las aplicaciones han de correr en la máquina virtual Dalvik para garantizar la seguridad del sistema.

9.1 Patrón Diseño

Android usa el patrón de diseño de Modelo Vista Controlador (MVC), este modelo separa de forma lógica los datos de una aplicación, la interfaz de usuario y la lógica de negocios en tres componentes distintos que se relacionarán para al final tener como resultado nuestra aplicación.

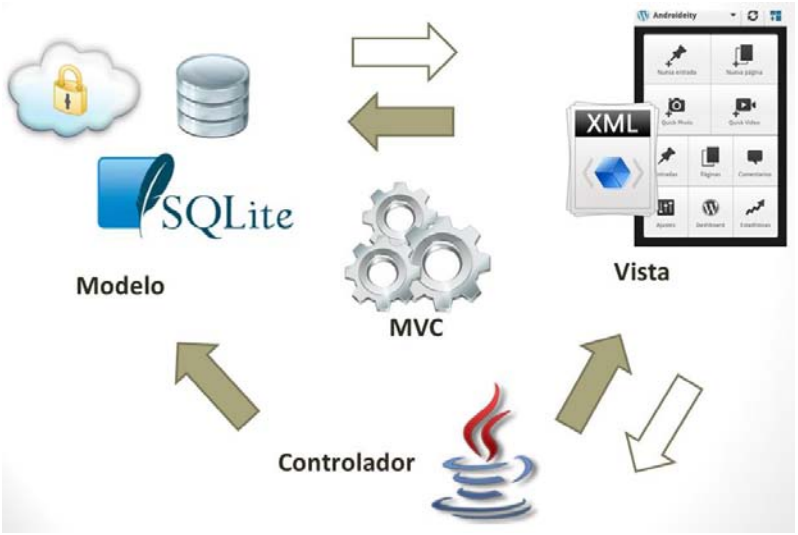


FIGURA 33

- **Modelo:** Es la representación que construirás basadas en la información con la que operará tu aplicación. En Java el modelo viene siendo análogo a los *beans* que tienen la particularidad de ser reutilizables y te ayudan a cumplir con el **proverbio de oro "Don´t Repeat Yourself" (DRY)** al hacer escalables tus aplicaciones. En esta parte del modelo hay que tener en cuenta donde vas a almacenar los datos de la información, en una Base de datos o un *Web services*.

- **Vista:** la vista es la interfaz con la que interactúa el usuario. En Android las interfaces se construyen con XML, tanto el esqueleto de la vista como los estilos se escriben en XML.

- **Controlador:** Responde a eventos e invoca peticiones al "modelo" cuando se hace alguna solicitud sobre la información. Son todas esas clases que ayudan a darle vida las interfaces y permiten desplegar y consumir información de o para el usuario. Estos controladores se programan en lenguaje Java y son el *core* de la aplicación.

MVC nos da la ventaja de poder separar el trabajo, de esta forma se puede separar la construcción en cada uno de estos componentes, obteniendo un resultado de calidad en cada uno y al final obtener una buena aplicación.

Con esta arquitectura en la aplicación se evita la multiplicidad de líneas de código que hacen lo mismo, es escala fácilmente la aplicación, se pueden agregar funcionalidades nuevas de manera más sencilla y facilitar el comienzo de nuevos proyectos.

A continuación se explica un ejemplo de cómo interactuará la aplicación poniendo el ejemplo de enviar un Tweet.

a) El controlador recibe la notificación de enviar el tweet. Por medio de un handler verificará si el usuario está logueado o no en su cuenta de Twitter.
b) El modelo es llamado para ser modificado. Se puede acceder al caché de tweets que se tenían desde la última vez que el usuario abrió la aplicación y se conectó a Internet y agregar el nuevo tweet.

c) Una vez que se tienen los nuevos tweets y la publicación que quiere hacer el usuario, el controlador nuevamente toma partida para llamar a la vista correcta que desplegará el Timeline actualizado.

10 DESARROLLO

10.1 Diseño

Para la realización inicial de la interfaz del usuario se realizaron mock-ups para definir las interacciones principales del usuario. En la realización de los Mock-ups se intentó definir una interfaz que sea usable y fácil de utilizar. Con este Mock- up tenemos un boceto preliminar de cómo sería la interfaz de la aplicación y la navegación de ésta entre las diferentes pantallas y secciones de nuestra aplicación. De esta forma conseguimos una primera visión general de todo el proceso para poder dividir mejor tareas, detectar errores en el diseño y poder incorporar cambios de forma sencilla, así como ver el impacto en el resto de la aplicación y su forma de trabajar con ella.

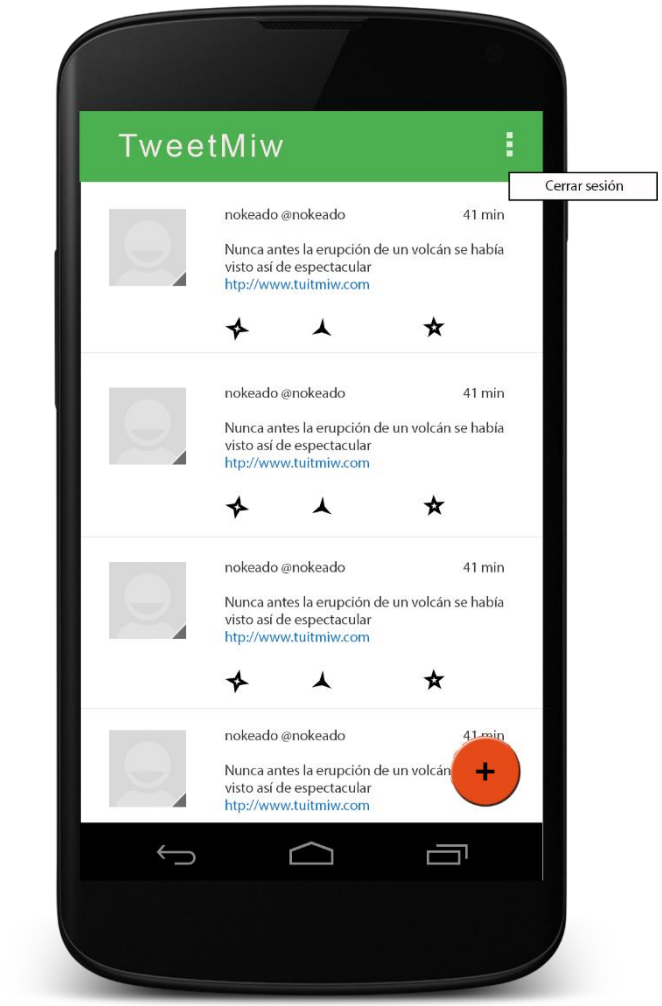


FIGURA 34: BOCETO DE LA APLICACIÓN

Una vez realizado el boceto inicial y puesto en común las ideas, comenzamos a implementar la aplicación y comenzamos a darle forma. Lo primero que nos planteamos es que queremos un proyecto moderno que utilice la última tecnología disponible, por lo que como ya hemos comentado anteriormente, nos decidimos por utilizar Android Studio, que en estos momentos es el IDE oficial para desarrollar en Android, y utilizamos Android 5.1.1 Lollipop (API 22). Para la parte de diseño nos decidimos por implementar Material Design en nuestro proyecto.

Una vez comenzamos a desarrollar la primera pantalla en Android Studio, nos planteamos la forma en la que vamos a realizar las diferentes pantallas. En un primer momento pensamos en hacerlo a través de iconos en la parte superior (al lado del menú desplegable), pero no nos acababa de convencer, y observamos que tanto la aplicación de Facebook como la de WhatsApp no están realizadas de esa forma., por lo que finalmente nos decidimos por utilizar un sistema de Sliding Tabs y fragments con el que poder desplazarnos entre las pantallas.

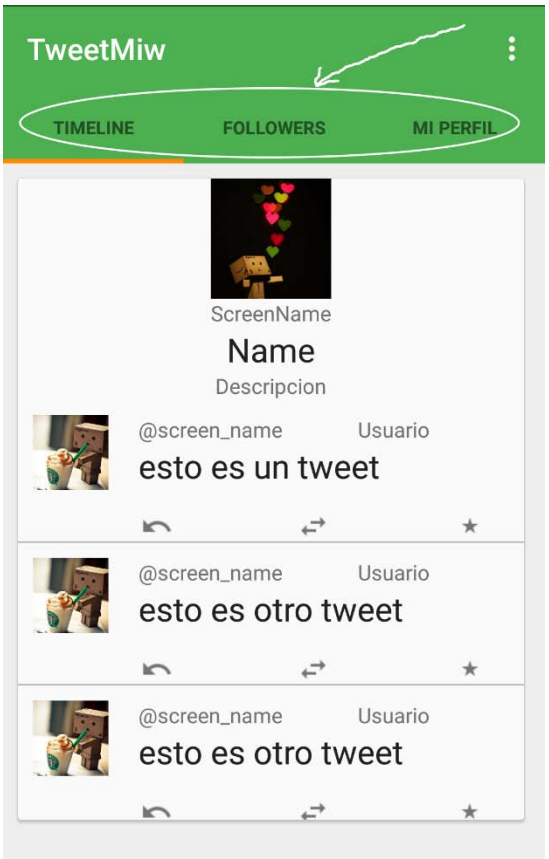


FIGURA 35: CAPTURA DEL DISEÑO DE LA APLICACIÓN

Llevamos a cabo esta implementación y empezamos a diseñar cada una de las pestañas para poder ir rellenándolas después. A partir de aquí, comenzamos con lo que es el desarrollo en cuanto a programación para realizar las funcionalidades que queremos que realice la aplicación y conforme lo vamos desarrollando, vamos mejorando el diseño,

intentando tener siempre en cuenta las recomendaciones de Material Design en cuanto a tamaño de los márgenes, botones, tamaño de letra, etc.

10.2 Implementación

Esta sección de la memoria pretende ser una guía para desarrolladores principiantes que quieran desarrollar una aplicación con Android.

10.2.1 Autenticación en Twitter con Fabric

Para poder conectar con Twitter es necesario asociar esta cuenta a una cuenta de Twitter, tras realizar este paso Twitter generará un consumer Key y un consumer Secret. Estas claves serán necesarias para poder autenticar nuestra aplicación desde Android. Para realizar el registro de la aplicación se puede realizar desde la propia cuenta de Twitter en la sección de developers. Sin embargo, para este proyecto se utilizó Fabric, framework propio de Twitter.

Fabric [6] es el framework de Twitter que permite desarrollar aplicaciones móviles, está desarrollado para facilitar el desarrollo móvil y hacerlo más ágil. Incluye reportes de fallos, distribución beta, análisis del comportamiento en los usuarios que acceden a una aplicación desde sus dispositivos móviles, entre otros.

Para poder hacer uso de Fabric es necesario tener los siguientes requisitos:

1. Crear un proyecto Android
2. Tener una cuenta de Twitter.
3. Tener una cuenta en Fabric.
4. Instalarse el plugin de Fabric en Android Studio.

Con el plugin instalado podremos realizar el proceso de login de la aplicación. El proceso de login con Fabric automáticamente modifica el proyecto en Android studio asociando las librerías en el archivo gradle, añadiendo en la actividad principal el consumer key y el consumer secret y modificando el Android manifest, añadiendo los permisos necesarios. Cabe destacar que con esta implementación realizamos la autenticación de la aplicación pero no del usuario

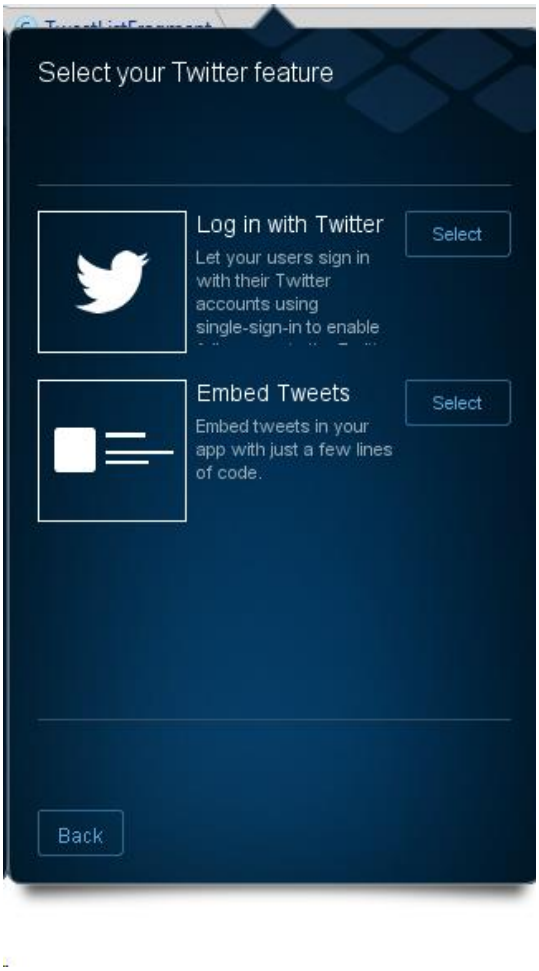


FIGURA 36

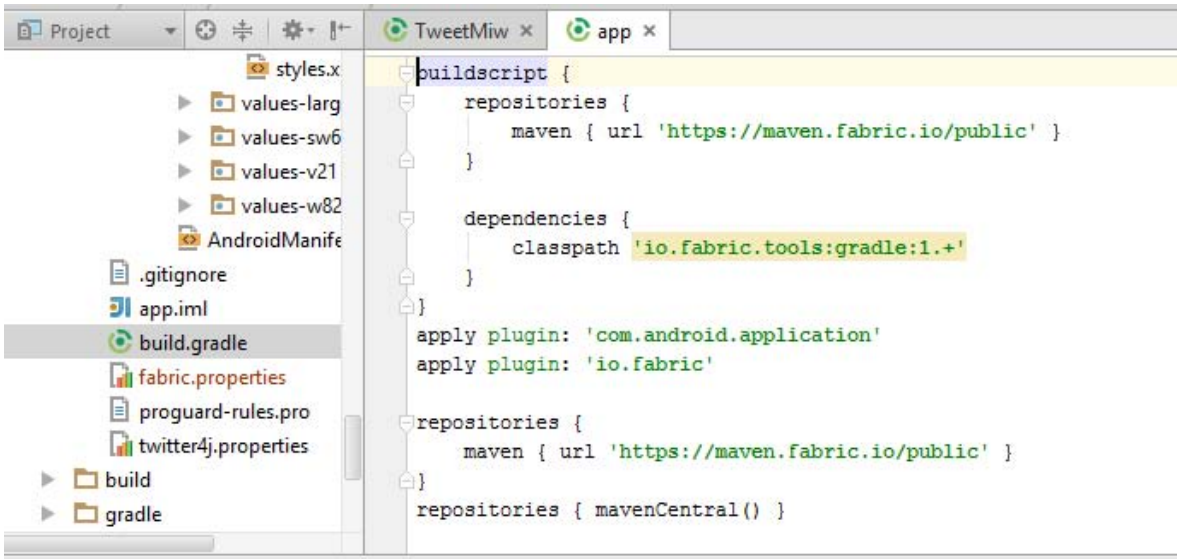


FIGURA 37

Para poder realizar la autenticación del usuario, Fabric redirige al usuario a la propia aplicación de Twitter para que el usuario autorice el uso de sus datos por nuestra aplicación.

Nuestra aplicación guardará dos elementos obtenidos cuando el usuario autorice el acceso, estos datos son outh_token y outh_token_secret, que serán necesarios para realizar las llamadas a la API. Para poder realizar este proceso necesitaremos los siguientes elementos:

1. Layout con bóton de Login de usuario
2. Actividad que realiza el tratamiento cuando detecta el evento onclick sobre el botón del Twitter.

Con esta modificación ya tendremos la aplicación lista para utilizar los recursos de Twitter. Cuando obtenemos el outh_token y el outh_token_secret se tendrá que almacenar estos datos para realizar las siguientes llamadas a la aplicación de Twitter.

Para almacenar estos datos se hizo uso de las preferencias de configuración de la aplicación (SharedPrefrences). Para realizar este almacenamiento se ha creado la clase SessionManager, dentro de esta clase se elige el método de seguridad para el acceso a estos archivos de configuración. Las posibilidades son:

- ***MODE_PRIVATE:*** sólo la aplicación podrá leer y escribir datos de configuración.
- ***MODE_WORLD_READABLE:*** sólo la aplicación podrá escribir datos de configuración, el resto de aplicaciones del dispositivo Android podrá leer los datos de configuración pero no modificarlos.
- ***MODE_WORLD_WRITEABLE:*** todas las aplicaciones del dispositivo Android podrán leer y escribir datos de configuración.
- ***MODE_MULTI_PROCESS:*** no suele usarse, puede servir cuando la aplicación tiene múltiples procesos, para "comunicación" entre ellos.

Los métodos que proporciona la clase SharedPreferences para leer los datos del fichero de configuración son:

- ***contains:*** comprueba si existe una preferencia en el fichero.
- ***getAll:*** obtiene todos los valores de todas las preferencias guardadas en el fichero.
- ***getBoolean:*** obtiene una preferencia de tipo Booleano (true, false).
- ***getFloat:*** obtiene una preferencia de tipo Float (numérico).
- ***getInt:*** Contiene una preferencia de tipo Int (entero).

- ***getLong:*** obtiene una preferencia de tipo Long (entero largo).
- ***getString:*** obtiene una preferencia de tipo String (texto).
- ***getStringSet:*** obtiene una preferencia de tipo StringSet (conjunto de cadenas).
- ***registerOnSharedPreferenceChangeListener:*** registra una devolución de llamada que se invoca cuando ocurre un cambio en las preferencias.
- ***unregisterOnSharedPreferenceChangeListener:*** anula el registro de una devolución de llamada anterior.

Según las preferencias que se deseen en la aplicación se definirán las distintas propiedades, en nuestro caso hemos definido el acceso en MODE_PRIVATE. Para guardar los valores de configuración en un fichero de preferencias de una aplicación Android los métodos que proporciona la clase SharedPreferences para guardar los datos son:

- ***apply:*** guarda los cambios realizados en las preferencias cuando el objeto ***SharedPreferencias*** está en modo edición.
- ***clear:*** elimina todos los valores de las preferencias.
- ***commit****:* guarda los cambios realizados en las preferencias.
- ***putBoolean****:* guarda un valor de tipo booleano (true, false).
- ***putFloat****:* guarda un valor de tipo Float.
- ***putInt****:* guarda un valor de tipo Int.
- ***putLong****:* guarda un valor de tipo Long.
- ***putString****:* guarda un valor de tipo String.
- ***puntStringSet****:* guarda un valor de tipo StringSet.
- ***remove:*** elimina el valor de una preferencia.

10.2.2 Realizar llamadas a la API de Twitter con Twitter4J

La implementación a la API REST de Twitter se realizó usando la librería de Twitter4J [7], esta es una librería de código abierto que permite realizar llamadas a la aplicación de Twitter con Java de forma más sencilla. Para realizar la conexión hay que crear un Token de acceso con los valores que ya hemos almacenado mediante la clase SessionManager (outh_token, outh_token_secret). Como vamos a realizar llamadas desde

varias clases de java, se crea un método en la clase SessionManager que devolverá la conexión en Twitter para realizar las llamadas al api.

```
/**
 * Metodo que ddevuleve una nstancia de Twitter del usuario logeado
 * @return twitter
 */
public Twitter getTwitter() {
    ConfigurationBuilder builder = new ConfigurationBuilder();
    builder.setOAuthConsumerKey(ConstantsUtils.CONSUMER_KEY);
    builder.setOAuthConsumerSecret(ConstantsUtils.CONSUMER_SECRET);
    String token = pref.getString(TOKEN, null);
    String secret = pref.getString(SECRET, null);
    AccessToken accessToken = new AccessToken( token,  secret);
    return new TwitterFactory(builder.build()).getInstance(accessToken);
}
```

FIGURA 38

Para poder completar la lista de requisitos que se desea implementar en el proyecto ha sido necesario realizar una llamadas a los siguientes recursos de la API REST de Twitter

Twitter4J	**Api Rest Twitter**	**Límite de llamadas**
getUserTimeline()	statuses/user_timeline.json	180
showUser(userId)	users/show.json	180
getHomeTimeline()	statuses/home_timeline.json	10
getFollowersList(userId,cursor)	followers/list.json	15
getFriendsList(userId,cursor)	friends/list.json	15
updateStatus(status)	statuses/update.json	1000/day
getFavorites()	favorites/list.json	15

10.2.3 Uso de tabs

Para la vista de la iteración principal del usuario se realizó el uso de Tabs [8] , ya que, esta vista hace más fácil explorar y cambiar entre las diferentes vistas de una aplicación. El único inconveniente en usar este tipo de navegación es que como máximo se tendrán 3 ó 4 tabs en la parte de la aplicación principal. El uso de Tabs exige que sólo la región seleccionada debe de cambiar el contenido a la hora de seleccionar una pestaña, y los indicadores de los otros Tabs deben permanecer en todo momento accesible. Además, la selección de Tabs no debe almacenarse como histórico, es decir, si un usuario cambia de una pestaña “A” a la “B”, al pulsar el botón Atrás no debe volverse a seleccionar la pestaña A.

Por último, y lo más importante, pestañas siempre deben correr a lo largo de la parte superior de la pantalla, y no deben estar alineados a la parte inferior de la pantalla. Las ventajas que tiene el uso de Tabs son las siguientes:

1. Los usuarios pueden navegar rápidamente entre las pantallas correspondientes, sin necesidad de volver a examinar la ventana padre.
2. Los usuarios tienen acceso inmediato a las distintas pantallas desde la pantalla principal.

Para poder añadir los Tabs es necesario asociar las dependencias necesarias en el gradle y añadir las siguientes clases que obtenemos de Google [9] SlidingTabStrip.java y SlidingTabLayout.java

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22.2.0'
    compile 'com.android.support:support-v4:22.2.0'
```

FIGURA 39

Para la implementación de los Tabs haremos uso de dos recursos ViewPager y ViewPagerAdapter. ViewPager asocia cada página con una única clave asociada a un objeto, en este caso Tab , esta clave única va a ser la que identifique el tab dentro del adaptador para nuestra vista, que en este caso nuero adaptador para las vistas es el ViewPagerAdapter.

Dentro del ViewPagerAdapter realizaremos un constructor dónde le pasaremos como parámetro el número e Tabs a implementar y el nombre de los Tabs. La asignación de cada uno de los nombre de los fragmentos se implementa en el método getPageTile. Hasta ahora tenemos la vista con nuestro Tabs vacíos. Sin embargo, el objetivo es que la aplicación muestre un fragmento en cada uno de los Tabs, para asignar que fragmento se van a cargar en cada vista se implementará el método getItem del adaptador.

```
@Override
public Fragment getItem(int position) {

    if(position == 0){ // if the position is 0 we are returning the First tab
        return TweetListFragment.init(position);
    }
    if(position == 1){ // if the position is 1 we are returning the Second tab
        return new Usuarios_Fragment();
    }
    else             // As we are having 2 tabs if the position is now 0 it must be 1 so
    {
        return new Tweet_Fragments();
    }

}
```

FIGURA 40

El objetivo final es tener una lista de fragmentos en cada uno de los Tabs que vamos a cargar, estos elementos son fragmentos que vamos a cargar en cada vista y cada fragmento tiene que estar asociado a un Adaptador y estos adaptadores son los que se encargarán de dibujar cada uno de los elementos de las listas.

10.2.4 Fragmentos

Un fragmento representa una porción de interfaz en una Actividad puede añadirse o eliminarse de una interfaz de forma independiente al resto de elementos de la actividad. La ventaja de usar fragmentos es que se puede reutilizar en varias actividades. Para utilizar un fragmento lo que se hace es añadir un Fragmento a la Actividad, que es el que muestra el layout que queremos mostrar. Estas características dotan a las aplicaciones Android la capacidad para adaptarse y responder a la interfaz de usuario sin importar el dispositivo.

Un fragmento siempre debe de ir asociado a una actividad, un fragmento que está asociado a una actividad, está asociado inherentemente a su ciclo de vida, es decir, cuando la actividad se destruye, todos los fragmentos asociados a esta actividad se destruirán.

La ventaja principal que tiene el uso de fragmentos comparado con lanzar varias actividades, es que dentro de un solo Activity podemos lanzar varios Fragments que interaccionan entre ellos a través de su Activity, de este modo se ahora el estar constantemente cambiando entre distintas actividades, con el consecuente gasto de memoria.

Para crear un fragmento debemos crear una subclase de Fragment, esta nueva actividad contiene método parecido a una clase los cuales son onCreate(), onStart(, onPause(), y onStop() . Los principales método que debemos implementar cuando estamos desarrollando la clase Fragment son los siguientes:

- onCreate(), El sistema lo invoca cuando se está creando el fragmento, aquí es donde tenemos que crear los componente del fragmento.
- onCreateView(), el sistema llama a este método cuando se quiere dibujar la interfaz del fragment, por lo que, este método debe devolver una vista.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    session = new SessionManager(getActivity());
    return inflater.inflate(R.layout.fragment_profile, container, false);
}
```

FIGURA 41

Para poder crear la vista del fragment es necesario implementar el método onCreateView(). Dentro de la implementación de este método se pasan tres parámetros de tipo LayoutInflater, ViewGroup y Bundle. El LayoutInflater es el elemento que ayuda a inflar o crear la vista del fragmento en la actividad. ViewGroup contienen la vista padre donde se va a insertar el nuevo fragmento. Finalmente el Bundle contiene información sobre una instancia previa del fragmento, por ejemplo ha estado en un estado resumed, entre otros.

Para poder cargar el nuevo fragmento en las vista hay que hacer uso del método inflate(), a este método se le pasan tres argumentos:

- El id del recurso que se quiere visualizar, el cual se puede obtener accediendo al recurso R.
- El ViewGroup, vista padre donde se quiere añadir el nuevo fragment
- Variable boolean ue indica si el Nuevo fragmento debe de ser incrustado a la nueva vista, si tenemos el fragmente dentro del Layout no es necesario realizar esto por lo que la variable tiene que tener valoro false. En el caso de tener el fragmento y la vista padre en dos layout, el valor debería ser true.

Para aclarar en qué momento se crea el fragmento o como interactúa dentro de una actividad se muestra un gráfico de como es el ciclo de vida de un fragmento mientras se está implementando en una actividad.

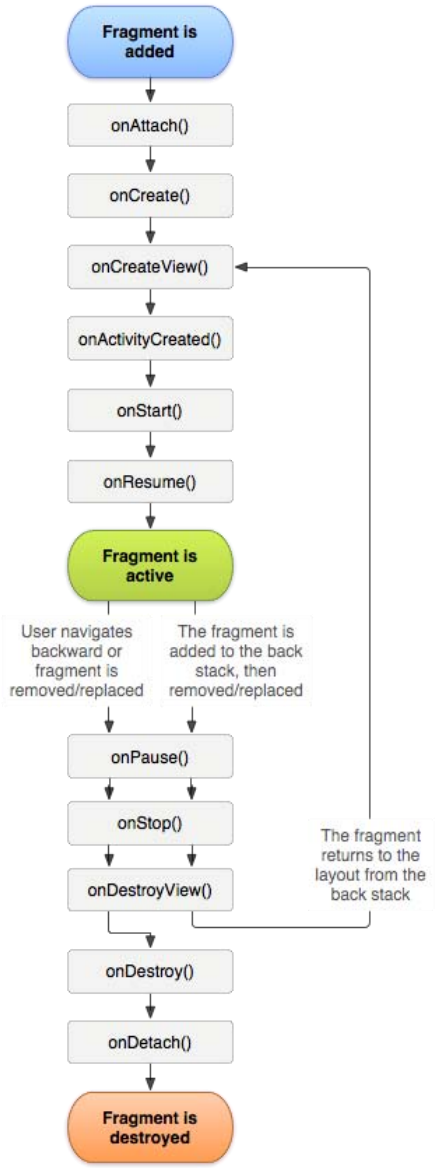


FIGURA 42

Además de usar fragmentos, para tener asociada una lista con fragmentos se puede hacer uso de la clase ListFragment, este elemento maneja una lista similar al ListActivity y contiene evento como onListItemClick(), el cual es de gran ayuda cuando queremos obtener

los datos específicos de una solo fragmento dela lista. Por ejemplo, cuando queremos ver más detalles de un usuario.

Uno de los aspectos de los que también se encargará el fragmento es crear la lista con los datos que se van a mostrar en pantalla. Esta lista de dato se los pasará como parámetro el fragmento a los adaptadores, los cuales se encargarán de asociar cada elemento de la lista a una vista.

10.2.5 Adapters

Un Adaptador es un puente entre los fragmentos y las vistas que queremos cargar. El adaptador permite el acceso a los elementos de datos, éste también es responsable de crear una vista para cada elemento en la colección de datos.

Se puede decir, que los adaptadores son colecciones de datos, que asignamos a una vista para que ésta los muestre, para este proyecto usamos adaptadores de tipo RecyclerView.Adapter. RecyclerView es la versión avanzada de ListView. Este nuevo contenedor permite cargar una larga lista de datos de manera eficiente.

Para poder definer este Adaptados es necesario sobre escribir dos métodos onCreateViewHolder para poder inflar la vista y su contenido y onBindViewHolder para asociar los datos de la lista a las vista

10.2.6 Asyntask

Android ejecuta todos sus componente en el mismo hilo, tanto servicios como. Es por ello, que cualquier operación larga o costosa que realicemos en este hilo va a bloquear la ejecución del resto de componentes de la aplicación y la interfaz. El usuario va a detectar estos problemas en la aplicación de forma que la aplicacion será lenta, se bloqueará, o tendra mal funcionamiento en general. Además, si la aplicación se bloquea por más de cinco segundos Android lanzará un mensaje "*Application Not Responding*" , en este caso el usuario decidirá entre forzar el cierre de la aplicación o esperar a que termine.

Las tareas asíncronas de Android proporcionan a Android la forma de crear tareas en segundo plano sin necesidad de tener que manejar hilos de ejecución

Las tareas asíncronas son de tres tipos: Params, Progress y Result:

- ***Params,*** parámetros enviados a la tarea en ejecución
- ***Progress,*** el progreso de la actividad que se está ejecutando en segundo plano
- ***Result,*** resultado de

No todos los parámetros se usan cuando se define la área asíncrona, si no queremos utilizar estos parámetro bastaría con remplaza su valor con un void. Además, las tareas asíncronas vienen definidas por cuatro métodos onPreExcute, doInBackground, onProgressUpdate y onPostExecute.

- ***onPreExecute(),*** este método se ejecuta antes de que la tares sea ejecutada, este método sirve para inicializar los parámetro, url, entre otros para poder ejecutar la tarea. En este método se suele implementar el progress bar que indica que se está ejecutando una acción en background
- ***doInBackground(Params...),*** este método se ejecuta inmediatamente después de terminar el método onPreExecute() , aquí es donde se ejecuta la operación que puede tardar varios segundos.
- ***onProgressUpdate***(Progress...), este método es ejecutado para mostrar cualquier progreso sobre la actividad que se está ejecutando en Segundo plano. Por ejemplo cuando estamos descargando un fichero nos muestre la cantidad de byte que están descargados.
- ***onPostExecute(Result),*** en este método es dónde recibiremos el resultado de la ejecución en segundo plano.

En este proyecto, se utilizaron tareas asíncronas para poder mostrar la lista de followers, ya que, en esta vista era necesario cargar las imágenes de los perfiles de los usuarios.

10.2.7 Buenas prácticas implementadas

Todo programador tiene que tener cierto cuidado con la calidad del código, para poder tener una calidad mínima del código se han seguido las siguientes pautas en el desarrollo de la aplicación.

- Evitar concatenaciones de Strings, utilizar StringBuffer o StringBuilder para tal caso. La concatenación de Strings produce cada vez un nuevo objeto y por tanto, mayor consumo de memoria y mayor recolección de basura.
- Crear métodos con el menor número de parámetros posible. Esta es de primero de SOLID.
- Sacar fuera de los bucles las constantes y creación de nuevos objeto
- Siempre que vayamos a acceder a la misma posición de un array varias veces, es mejor guardar esa posición en una variable local y así evitar el acceso repetido al índice del array.
- Utilizar buffers para leer datos a través de la red y leer los datos en porciones en lugar de byte a byte que es más lento.
- Referenciar a null instancias de objetos que ya no se van a usar, para que el recolector de basura libere memoria.
- El acceso a los atributos de una clase es más rápido que encapsular con getter y setter.
- El acceso a variables locales es más rápido que a atributos de la clase. Siempre que sea posible asignar atributos de una clase a una variable local si se va a hacer referencia a ella varias veces dentro de un método o bucle.
- Usar operadores como x+=1 en vez de x = x+1 ya que generan menos byte code.
- Cuando sea posible evitar bucles ya que evitaremos toda la sobrecarga de control de flujo en cada iteración. Por ejemplo, si tenemos una operación que se va a realizar 5 veces, en vez de utilizar un bucle podemos realizar las 5 operaciones secuencialmente.
- Usar excepciones únicamente cuando sea necesario ya que cada excepción lanza un nuevo objeto.

11 PRUEBAS

Las pruebas que se realizaron en Android Studio se realizaron utilizando el componente de Activity Testing [14] de Android. Las pruebas se han desarrollado sobre las entidades que hemos definido en nuestra carpeta de entities. La implementación de pruebas en Android se desarrolló basándose en JUnit. Android Studio por defecto crea un paquete de pruebas en el directorio src/androidTest/java, dentro de este paquete el programador creará todas las clases de Test que desee.

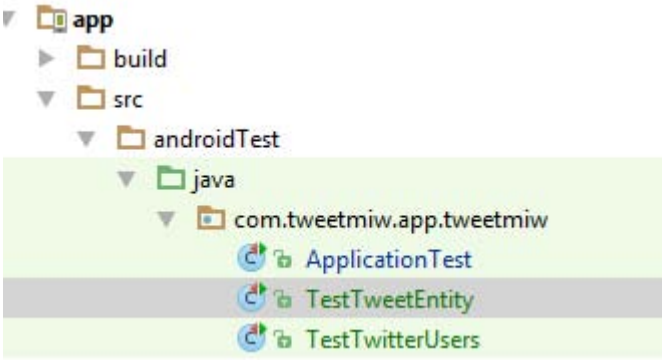


FIGURA 43

Antes de empezar a definir las pruebas, es necesario definir en el archivo graddle dentro de defaultConfig, dónde se van a implementar los test (testApplicationId), definir testInstrumentationRunner, herramienta que sirve para arrancar los test y que extiende de JUnit y definir el directorio dónde se guardan los resultados de los test.

```
defaultConfig {
    applicationId "com.tweetmiw.app.tweetmiw"
    minSdkVersion 16
    targetSdkVersion 22
    versionCode 1
    versionName "1.0"
    testApplicationId "com.tweetmiw.app.tweetmiw"
    testInstrumentationRunner "android.test.InstrumentationTestRunner"
}
testOptions {
    reportDir = "$project.buildDir/results/report"
    resultsDir = "$project.buildDir/results"
}
```

FIGURA 44

Para poder ejecutar los test, es necesario tener un dispositivo conectado para poder ejecutar las pruebas. Dentro de las clases de pruebas podemos sobrescribir dos métodos que se ejecutarán antes y después de cada pruebas, setUp antes y tearDown() después.

Con la implementación de pruebas nos aseguramos que los pequeños módulos de nuestra aplicación funcionan sin necesidad de tener que ejecutar toda la aplicación.

12 GOOGLE PLAY

Google Play Store (anteriormente Android Market) es una plataforma de distribución digital de aplicaciones móviles para los dispositivos con sistema operativo Android, así como una tienda en línea desarrollada y operada por Google. Esta plataforma permite a los usuarios navegar y descargar aplicaciones (desarrolladas mediante Android SDK), juegos, música, libros, revistas y películas. También se pueden adquirir dispositivos móviles como ordenadores Chromebook, teléfonos inteligentes Nexus y Google Chromecast, entre otros.

FIGURA 45: LOGO DE GOOGLE PLAY

Las aplicaciones se encuentran disponibles tanto de forma gratuita como de pago y pueden ser descargadas directamente desde un dispositivo con Android a través de la aplicación móvil Play Store. Es posible también instalar estas aplicaciones directamente y sin necesidad de una computadora, en dispositivos con sistema operativo BlackBerry 10.

Los usuarios también pueden instalar aplicaciones a través del sitio web del desarrollador o mediante tiendas de distribución digital alternativas. Las aplicaciones pueden ser actualizadas automáticamente si el usuario así lo establece, o pueden hacerlo manualmente una a una.

El 17 de marzo de 2009, alrededor de 2.300 aplicaciones estaban disponibles en Android Market, de acuerdo con el director técnico de T-Mobile Cole Brodman. El 10 de mayo de 2011, durante Google I/O, Google anunció que en Android Market figuran 200.000 aplicaciones y habían sido instaladas 4.500.000.000 aplicaciones.

En marzo de 2012, con la fusión de Android Market con Google Music, el servicio fue renombrado a Google Play, como resultado de la nueva estrategia de distribución digital de Google. En julio de 2013, se anunció que Google Play había sobrepasado 1 millón de aplicaciones publicadas y se habían registrado más de 50 mil millones de descargas.

Las aplicaciones desarrolladas por Google para utilizar la plataforma Google Play y completar el entorno, son:

- ***Play Libros:*** Es una aplicación móvil que permite leer libros electrónicos. Fue desarrollada por Google en febrero de 2011. Play Libros ofrece más de 4 millones de títulos que pueden ser adquiridos a través de Google Play. Los libros adquiridos son almacenados en la nube, logrando con esto estar disponibles para acceder a ellos directamente en esta aplicación o a través de la web. Los usuarios pueden almacenar más de 1,000 archivos sin costo, mientras cada archivo no supere los 50 MB de espacio.
  El 15 de mayo de 2013, se actualizó Play Libros añadiendo la posibilidad de que los usuarios pudieran subir archivos en formato PDF y EPUB. Play Libros se encuentra actualmente disponible en 44 países.

- ***Play Juegos:*** Es un servicio y aplicación desarrollado para Android, iOS y web que añade la opción de multijugador en tiempo real, logros, tabla de posiciones y guardar información en la nube de los juegos que sean compatibles con este servicio. El servicio fue presentado en el Google I/O de 2013, y la aplicación fue lanzada el 24 de Julio de ese mismo año en el evento llamado "Breakfast with Sundar Pichai".

- ***Play Kiosco:*** Google Play ofrece la posibilidad de suscribirse a revistas y diarios de noticias en algunos países, entre los que están Estados Unidos, Australia, Canadá y el Reino Unido, esto a través de las aplicaciones Google Currents y Play Magazines. El 20 de noviembre de 2013, Play Magazines y Google Currents fueron combinadas dando lugar a Play Newsstand (Play Kiosco en los países de habla hispana), aplicación que combina los servicios de ambas aplicaciones en un solo producto.

- ***Play Películas:*** Es una aplicación que permite ver películas y series de televisión adquiridos a través de Google Play. Las películas pueden comprarse o alquilarse, mientras que las temporadas o capítulos de las series no están disponibles para alquilar, sólo se pueden comprar. Los usuarios tienen la posibilidad de descargar el contenido para poder verlo

posteriormente sin necesidad de una conexión a Internet. Play Películas se encuentra disponible en 25 países, pero las series sólo están disponibles en Estados Unidos, Japón y el Reino Unido.

- ***Play Música:*** Es un servicio de almacenamiento y sincronización de música en la nube, así como también tienda musical en línea, lanzada en 2011 tras haberlo anunciado oficialmente durante la conferencia de desarrolladores Google I/O del 10 de mayo de 2011. El servicio permite el almacenamiento gratuito de canciones propias hasta un total de 50.000, que al sincronizarse en la nube, se podrá acceder desde cualquier punto. También permite escuchar música a través de streaming, servicio el cual se ofrece mediante suscripción.

  El servicio, es accesible vía navegador, cliente de escritorio y smartphones con Android, aunque también se puede acceder desde otros sistemas operativos móviles siempre y cuando soporten Adobe Flash.

12.1 Clasificación de Aplicaciones

Google Play, al igual que su competidor App Store, tiene sistema de clasificación por edades pero desde el 8 de junio de 2015 las apps son clasificadas por sistemas de clasificación por edades de diferentes regiones y países, siempre y cuando no sean sobrescritas por normas como ESRB o PEGI.

Clasificación:

- ***Para todos:*** contenido sin restricciones de edad (Equivale a 4+ de App Store)
- ***Madurez baja:*** recomendado para mayores de 6 - 7 años (equivale a 9+ de App Store)
- ***Madurez media:*** Recomendado para mayores de 12 años (equivale a 12+ de App Store)
- ***Madurez alta:*** Recomendado para mayores de 17 años (equivale a 17+ de App Store)

12.2 Interfaz

Google Play tiene un acceso fácil y rápido a sus aplicaciones. Las aplicaciones son creadas por desarrolladores de todo el mundo y posteriormente puntuadas por los usuarios de Android.

Las categorías de juegos y aplicaciones, del menú principal proporcionan submenús para que la búsqueda sea más sencilla. Los usuarios tienen la posibilidad de valorar las aplicaciones mediante un sistema similar a YouTube, con una escala del 1 al 5, ofreciendo también la posibilidad de poner comentarios sobre la aplicación. La novedad de la nueva versión de Google Play es también la posibilidad de añadir capturas de pantalla de su aplicación.

12.3 Desarrolladores

La gran novedad que aporta Google Play hace referencia a los desarrolladores: estos serán capaces de hacer su contenido disponible en un servicio abierto que ofrece una retroalimentación y sistema de calificación similar a YouTube. Los desarrolladores tendrán un entorno abierto y sin obstáculos para hacer su contenido disponible. El contenido puede subirse al mercado después de tres pasos: registrarse como comerciante, subir y describir su contenido y publicarlo. Para registrarse como desarrollador y poder subir aplicaciones hay que pagar una cuota de registro (US$ 25,00) con tarjeta de crédito.

Los desarrolladores de las aplicaciones de pago reciben un 70% del precio total de la aplicación, mientras que el 30% restante es destinado a las empresas. El beneficio obtenido de Google Play es pagado a los desarrolladores a través sus cuentas en el sistema de Google Checkout.

12.4 Disponibilidad para Desarrolladores

En un primer momento sólo los desarrolladores en Estados Unidos y Reino Unido tenían soporte para publicar aplicaciones de pago. Actualmente Google ha aumentado esa lista considerablemente hasta alcanzar los 74 países. Por el contrario la lista de países con disponibilidad de desarrolladores que pueden distribuir aplicaciones gratuitas asciende a 152.

12.5 Competencia y Aliados

Las tiendas por Internet están cada vez más en alza, y más si se trata de tiendas de aplicaciones para los teléfonos móviles. No es de extrañar que las principales plataformas de móviles decidan crear portales donde poder descargar o comprar todas las aplicaciones posibles. Entre estas nos encontramos dos de las más importantes: la App Store de Apple y Google Play de Google; pero también hay otras como Samsung Apps de Samsung, Nokia Store de Nokia, App Place de Toshiba Market Place, App World de Blackberry, Windows Phone Store y Windows Store de Microsoft, App Store de Amazon, Palm App Catalog y SlideME.

El desarrollo de aplicaciones para el iPhone no está siendo de fácil acceso como consecuencia de la política de admisión de aplicaciones de Apple que es muy restrictiva. En cambio, Google Play no hace tantas excepciones con las aplicaciones, aceptando todas, propias o de desarrolladores, gracias a su herramienta Android SDK.

Por lo que se refiere al número de descargas, Google Play supera a App Store en número de descargas desde junio de 2013 y en 2014 consigue superar en número de aplicaciones ofrecidas desde Google Play a App Store, la tienda de Apple. En España la evolución de Google Play parece ser mejor comparada con el nivel mundial. Además se pueden instalar aplicaciones directamente en el dispositivo, si se dispone del archivo APK de la aplicación.

13 PUBLICACIÓN EN GOOGLE PLAY

13.1 Versión de la Aplicación

Las versiones en las aplicaciones Android se rigen por dos valores: Version code y Version name. El primero de ellos debe ser un número consecutivo, es decir, que la primera versión será la 1, la siguiente la 2, etc. El nombre de versión puede indicarse de cualquier manera, aunque lo habitual es usar dos o tres números separados por puntos, indicando el primero de ellos el número principal de versión (que cambia cuando hay cambios importantes en la aplicación, como un cambio completo de su interfaz), el segundo puede ser la subversión (con cambios menos importantes en la aplicación, como alguna nueva funcionalidad) y el tercero el número de compilación (para arreglos de pequeños errores, por ejemplo).

Estos dos datos se indican en las aplicaciones Android en el archivo build.gradle de la aplicación, dentro de android > defaultConfig, con los atributos versionCode y versionName.

13.2 Crear Firma Digital de la Aplicación

Para poder publicar nuestra aplicación en Google Play Store debemos firmarla digitalmente. Este proceso hay que realizarlo como medida de seguridad y como requisito de garantía, ya que de esta forma sólo nosotros podremos modificar y actualizar nuestra aplicación.

La primera vez que queremos publicar una aplicación debemos generar lo que se conoce como “Key store path”, la cual almacena nuestra firma. Este archivo se almacena en nuestro ordenador y hay que guardarlo con cuidado, puesto que si alguna vez perdemos el fichero, no podremos actualizar ni hacer más cambios en nuestra aplicación; deberemos volver a subir la aplicación y empezar de cero en cuanto a descargas, estadísticas, valoraciones, etc, lo cual sería un desastre.

Cada vez que vayamos a actualizar nuestra aplicación, tendremos que generar nuestra APK firmado con la firma digital creada anteriormente. La firma digital sólo se genera la primera vez y sirve para más de una aplicación.

13.3 Exportar la Aplicación

Una vez que la aplicación está suficientemente desarrollada y se considera que ya no está en proceso de depuración, se debe indicar al proyecto que pasa de la fase de depuración (debug) a la fase de lanzamiento (release). Si no se realiza este paso en el momento de intentar publicar la aplicación obtendrás un mensaje informando sobre ese problema.

El cambio de estado del proyecto se realiza desde la pestaña Build Variants que puedes encontrar en el margen izquierdo de Android Studio. También se ha de modificar el archivo build.gradle para activar ProGuard. Simplemente se ha de modificar la línea de "minifyEnabled false" y poner "minifyEnabled true". ProGuard sirve para optimizar, reducir y ofuscar el código de nuestras aplicaciones Android.

Para asegurarnos de que vamos a publicar la última compilación del proyecto conviene realizar una reconstrucción completa del proyecto desde el menú Build > Rebuil Project.

En el mismo menú anterior podemos ver la última opción: Generate Signed APK. Esa opción es la que usamos ahora para generar el archivo compilado y firmado, con la firma que hemos generado anteriormente.

FIGURA 46: CAPTURA DE PANTALLA DE LA FIRMA DE LA APLICACIÓN

Si realizamos todo correctamente, podremos encontrar el archivo APK en la carpeta del módulo creado en el proyecto. Ese es el archivo que vamos utilizar durante el proceso de publicación a través de la consola de desarrollador.

13.4 Consola de Desarrollador

La consola de desarrollador de Google Play es el lugar desde donde gestionaremos todas nuestras aplicaciones Android. Como ya hemos comentado anteriormente, para activar nuestra cuenta de desarrolladores de aplicaciones con Google, debemos pagar una cuota de 25$ que se paga sólo la primera vez y activa esa cuenta para siempre.

A través de la consola de desarrollador, podremos publicar y actualizar nuestras aplicaciones, agregar y modificar su información (como la descripción corta, la descripción larga, las imágenes, etc).

Dentro de la consola también podremos ver las estadísticas de nuestra aplicación, conocer cuánta gente se la ha descargado, leer las opiniones y valoraciones de la gente, configurar el precio de la aplicación (creando previamente una cuenta de comerciante), así como recibir notificaciones por parte de Google sobre mejoras que deberíamos implementar en nuestra aplicación.

13.5 Publicación de la Aplicación

Una vez activada la cuenta de desarrollador de aplicaciones de Google (abonando el dinero), resulta muy sencillo publicar una aplicación. Sólo deberemos hacer click en "Añadir nueva aplicación", subir nuestra APK firmada que hemos generado previamente a través de nuestro Android Studio (como se ha expuesto anteriormente) y seguir los pasos que se nos indican, como ponerle nombre a la aplicación y subir las imágenes con las resoluciones que nos solicitan para poder completar el aspecto visual de nuestra aplicación dentro de Google Play Store.

14 CONCLUSIONES

Con este proyecto hemos profundizado en la programación con Android enfrentándonos con todo el proceso de realización de una aplicación para móvil. Hemos intentado implementar las últimas funcionalidades y tecnología en el ámbito de Android usando la versión de la API más moderna, así como implementando las últimas recomendaciones de Google en el aspecto visual, como es Material Design.

También hemos profundizado y mejorado nuestro manejo en el Control de Versiones a través de Github, puesto que era la única forma de tener controlado siempre nuestro código.

Nos hemos encontrado con diferentes obstáculos que hemos tenido que ir solventando, como por ejemplo ha sido el tema de la distancia entre los dos componentes del grupo (vivimos en ciudades diferentes), lo que ha dificultó, sobre todo al principio, el tema de repartir las tareas, poner en común las ideas y realizar el diseño inicial.

Por otra parte, también nos hemos encontrado con una dificultad añadida, y es que como el ámbito de los dispositivos móviles evoluciona tan rápido y cada año sale, al menos, una versión nueva de Android os encontrábamos con información que, en muchos casos, estaba obsoleta o cuyo código nos daba “deprecated” cuando intentábamos implementarlo en nuestro código.

15 LÍNEAS FUTURAS

La aplicación que hemos creado cumple con unos requisitos básicos (que creemos que son los más utilizados por los usuarios) debido a la limitación de tiempo que hemos tenido para realizar el proyecto. Sin embargo, tenemos previstas diversas actualizaciones y mejoras que se podrían realizar en el proyecto:

- Mostrar la imagen del usuario que ha realizado el tuit.
- Implementar un apartado en el que poder visualizar un mapa con los tuits cercanos.
- Implementar un sistema de "tuits pendientes" con el que poder tuitear sin tener cobertura y que se publique una vez recuperemos los datos. Esto lo lograríamos guardando los datos de ese tuit en la caché del teléfono.
- Implementar la funcionalidad de buscar tuits por su hastag o a usuarios dentro de la aplicación.
- Mejorar la implementación de Material Design agregando animaciones a las vistas y a los botones.
- Actualizar la aplicación para la nueva versión de Android M, para poder ofrecer las nuevas funcionalidades, como serán el control de los accesos, etc.
- Cachear mejor la aplicación para que no tenga que hacer tantas llamadas a la API de Twitter y conseguir de esa manera una aplicación más fluida.

16 REFERENCIAS

[1] [En línea]. Available: http://www.gartner.com/newsroom/id/2875017.

[2] [En línea]. Available: http://es.wikipedia.org/wiki/Anexo:Historial_de_versiones_de_Android.

[3] [En línea]. Available: https://developer.android.com/about/dashboards/index.html?utm_source=suzunone.

[4] D. VM. [Online]. Available: http://en.wikipedia.org/wiki/Android_%28operating_system%29.

[5] [En línea]. Available: http://en.wikipedia.org/wiki/Dalvik_%28software%29.

[6] [En línea]. Available: https://fabric.io/.

[7] [En línea]. Available: http://twitter4j.org/en/index.html.

[8] [En línea]. Available: https://www.google.com/design/spec/components/tabs.html#tabs-usage.

[9] [En línea]. Available: https://github.com/google/iosched/tree/master/android/src/main/java/com/google/samples/apps/iosched/ui/widget.

[10] Wikipedia-JSON, http://es.wikipedia.org/wiki/JSON. [En línea].

[11] Gartner, «http://microsoft-news.com/gartner-microsoft-will-be-irrelevant-in-next-four-years-if-they-dont-succeed-in-tablet-market/,» [En línea].

[12] [En línea]. Available: http://androideity.com/2012/05/10/la-importancia-del-mvc-en-android/.

[13] [En línea]. Available: http://www.androidhive.info/2014/05/android-working-with-volley-library-1/.

[14] [En línea]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer.

[15] [En línea]. Available: https://github.com/.

17 BIBLIOGRAFÍA

- Libro de Desarrollo de Aplicaciones Móviles (Curso de Android) – Maestros del Web
- Guía Online de Material Design creada por Google
- Video Tutoriales de la Comunidad Platzi
- Estructura de un proyecto en Android Studio
    - http://www.hermosaprogramacion.com/2014/08/android-studio-proyecto-en/
- Material Design:
    - http://www.elandroidelibre.com/2014/11/que-es-material-design.html
    - https://es.wikipedia.org/wiki/Material_design
- Android M:
    - http://www.androidpit.es/android-m-fecha-lanzamiento-funciones
- Dart:
    - http://hipertextual.com/2015/05/dart-android
- Estados de una actividad:
    - https://luixrodriguezneches.wordpress.com/2012/10/28/estados-de-una-actividad-en-android/
- Google Play Store:
    - https://es.wikipedia.org/wiki/Google_Play
- Publicación en Google Play:
    - http://javiergarbedo.es/linux/90-guadalinex/356-preparar-aplicacion-para-publicarla-2
    - http://androideity.com/2011/08/25/%C2%BFcomo-firmar-aplicaciones-android/
    - https://miguelangellv.wordpress.com/2013/04/23/proguard-optimiza-reduce-y-ofusca-el-codigo-de-tus-aplicaciones-android/
- Estudio del estado del arte:
    - http://www.artyco.com/blog/uso-dispositivos-moviles-en-espana/
    - http://blogthinkbig.com/sistemas-operativos-moviles/
    - http://www.areatecnologia.com/informatica/sistemas-operativos-moviles.html